Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

REPRESENTACIÓN DE LOCUCIONES MEDIANTE EMBEDDINGS EXTRAÍDOS CON DNNS PARA RECONOCIMIENTO AUTOMÁTICO DE LOCUTOR

Autor: Marcos Asenjo González
Tutor: Alicia Lozano Díez
Ponente: Joaquín González Rodríguez

JUNIO 2019

REPRESENTACIÓN DE LOCUCIONES MEDIANTE EMBEDDINGS EXTRAÍDOS CON DNNS PARA RECONOCIMIENTO AUTOMÁTICO DE LOCUTOR

Autor: Marcos Asenjo González
Tutor: Alicia Lozano Díez
Ponente: Joaquín González Rodríguez

Audio, Data Intelligence and Speech - Audias
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JUNIO 2019

Resumen

Resumen

Este Trabajo de Fin de Grado tiene como objetivo la implementación y análisis de un sistema extractor de representaciones de locutores de una longitud fija (embeddings) a partir de audios de una duración variable para la posterior verificación de locutor. Con ello se busca profundizar en técnicas que ya han demostrado superar a los métodos más usados (como los modelos de variabilidad total, los i-vectors) y analizar qué configuraciones de parámetros del extractor obtienen mejores resultados.

Para abordar esto, se entrenará una red neuronal profunda con diferentes configuraciones de parámetros y tipos de arquitectura en la tarea de clasificación de locutores. Lo que se busca es que la red sea capaz de extraer aquellas características que mejor identifican al autor de una determinada locución (que serán luego las que se extraigan), y para ello emplea un entorno supervisado para aprender mediante descenso por gradiente cuáles son las transformaciones más adecuadas para este fin. Para poder conseguir esto, la red tendrá una parte dependiente del tiempo en la que se tratarán las dependencias temporales de las secuencias de audios y luego, tras una agrupación de información basada en estadísticos, habrá una parte no dependiente del tiempo, en el que se trabaja directamente sobre representaciones independientes de la longitud de los audios.

Además, en este trabajo se ha realizado un estudio sobre cómo la aplicación de modelos de análisis de factores (LDA y GPLDA) afecta a los resultados del sistema completo, y se analizan las ventajas e inconvenientes de la integración de estos modelos con el extractor implementado con respecto a la métrica de similitud coseno.

El conjunto de datos empleado en este trabajo ha sido una unión de los datos de las *"Speaker Recognition Evaluations"* de NIST de cinco años distintos (2004, 2005, 2006, 1008 y 2010), seleccionado por su popularidad y porque provee una gran cantidad de datos sin mucho ruido en sus etiquetas. Las principales herramientas con las que se trabajará son Keras y Tensorflow para la implementación del extractor y la obtención de los vectores representación, y Matlab para la aplicación de las técnicas de análisis de factores. Además, se cuenta con aportaciones en Kaldi.

En la parte final se presentan los resultados obtenidos al ir modificando distintos parámetros del extractor (número de neuronas por capa, número de capas, cantidad de datos empleados, longitud de los datos empleados, inclusión de técnicas contra el sobre-entrenamiento y distinta dimensionalidad en LDA y GPLDA). También se incluyen las conclusiones que se pueden extraer de los resultados presentados y las posibles líneas de trabajo futuro.

Palabras Clave

Reconocimiento automático de locutor, redes neuronales profundas, embeddings, BLSTM, RNN, TDNN, LDA, GPLDA, sistema independiente de texto.

Abstract

This Bachelor Thesis has as a purpose the implementation and analysis of an extractor system of fixed-length representations of speakers (embeddings) by using some variable duration audios for the subsequent speaker's verification. The aim is to delve into a series of techniques which have already proved to be superior than the most used methods (such as total variability models: i-vectors) and to sort out which extractor parameters settings get the highest results.

To address this, a deep neural net with diverse parameters settings and kinds of architecture in the task of classifying speakers will be trained. What is intended is the net to be capable of extracting those characteristics which most accurately identify a specific locution's speaker (which later will be the ones to be extracted). In order to do this the net uses a supervised environment to learn by gradient descent which are the most appropriate transformations for this purpose. Aiming to succeed in this, the net will be composed of a time-dependent part in which the audio streams temporary dependences will be exploited. Later, after a statistics information grouping there will be a time-independent part which will work directly on fixed-length audio representations.

In addition, a study about how the application of factor analysis models (LDA and GPLDA) affect the whole system's results has been carried out. Moreover, advantages and disadvantages regarding the integration of these models with the implemented extractor with respect to the cosine scoring metrics have been analysed too.

The dataset used in this project is the union of the NIST "Speaker Recognition Evaluations" data from five different years (2004, 2005, 2006, 2008 and 2010), selected by its popularity and the considerable amount of data it provides without much noise in its tags. The main tools used are Keras and Tensorflow for the implementation of the neural network and the extraction of embeddings, and Matlab for the application of factor analysis techniques. Besides, the project counts on some contributions using Kaldi.

In the final part the results obtained are presented when modifying different parameters of the extractor (number of neurons per layer, number of layers, the amount of data that has been used, data length, inclusion of techniques against overfitting and different dimensionality in LDA and GPLDA). Finally, the conclusions extracted from the obtained results are compiled along with some possible future work guides.

Key words

Automatic speaker recognition, deep neural networks, embeddings, BLSTM, RNN, TDNN, LDA, GPLDA, text-independent systems.

Agradecimientos

Quisiera agradecer en primer lugar a mi tutora Alicia Lozano por toda su ayuda a lo largo de este proceso y sin la que sin duda este trabajo no habría sido posible, pero sobre todo por ser la mejor tutora que uno podría desear. También a mi ponente, Joaquín González Rodríguez, por su paciencia y apoyo en los momentos de mayor duda. A Rubén Zazo, gracias al cual estoy donde estoy y soy lo que soy. También quisiera agradecer a mi familia y amigos el apoyo que me han brindado en estos cuatro años y en los que seguirán, y por estar ahí en todos esos momentos de cansancio y desesperación. A mis compañeros de AUDIAS, que me ayudaron con todo lo que estaba en sus manos de manera desinteresada y con una sonrisa, y siempre se preocuparon por mí. Y finalmente, a Clara, a Blanca y a Cano, por su ayuda en la redacción de estas páginas.

«Siempre listos»

Marcos Asenjo González

Junio 2019

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación

En los últimos años no sólo se ha mantenido la inmensa difusión iniciada la década pasada de los dispositivos portátiles y *"wearables"*, sino que se ha consolidado, cambiando la industria, la forma de relacionarnos con la tecnología y los desafíos y posibilidades que se nos plantean. Poco a poco estos dispositivos se han convertido en una extensión de nuestra identidad, asistentes personales no sólo capaces de reconocer el significado de lo que decimos sino también de identificarnos por nuestra voz y otros rasgos característicos. Además, la forma de comunicarnos con ellos cada vez más se basa en el habla más allá de la interacción con dispositivos como pantallas o teclados.

Por ello, la identificación y verificación de locutor ha ganado visibilidad y significatividad en los últimos tiempos, ya que el habla es una parte integral en la interacción hombre-máquina y sus aplicaciones han aumentado drásticamente. Así, cobra especial importancia para la autenticación biométrica, para la personalización de los servicios, para el etiquetado automático de locuciones, como prueba en casos penales, para seguridad nacional, para ciencias forenses y así muchos ejemplos más.

Además, esta popularización de los dispositivos portátiles ha traído consigo la disponibilidad de una mayor cantidad de grabaciones de audio y por tanto la creación de bases de datos más grandes y completas, lo que facilita la tarea de algoritmos de aprendizaje automático. En este sentido también hay que destacar la creación de entidades que buscan regularizarla y estandarizarla, como el *"National Institute of Standards and Technology"*.

Por otra parte, el aumento progresivo de la capacidad de cómputo, que hacía que los algoritmos más costosos fueran cada vez más asequibles, unido a los buenos resultados que han obtenido las redes neuronales profundas en otros ámbitos del procesamiento de audio y muy cercanos al reconocimiento de locutor (como el reconocimiento de idioma, de locuciones o la síntesis de voz), sitúan a estos modelos como aproximaciones muy interesantes para este tipo de problemáticas difíciles. Esto ha hecho que en los últimos años hayan sido objeto de investigación y se hayan realizado numerosos avances en éste ámbito. Así, ya desde 2012 se consiguen modelos con redes neuronales que mejoran el estado del arte anterior (basado en GMM, como se verá en 2.1.1) y a partir de entonces se encuentran entre los métodos más populares en todas las fases de los sistemas de reconocimiento de locutor (tanto en extracción de características, como en

tratamiento de ellas, o formando sistemas *"end-to-end"*). Dentro de este ámbito nos centraremos en los embeddings, que en los últimos años han superado a técnicas anteriores como los i-vectors y son especialmente interesantes por ser representaciones exitosas de locutor e idioma de una longitud fija.

Por todo esto, en este trabajo se emplea un modelo extractor de representaciones de locuciones basado en redes neuronales profundas, buscando aprovechar las ventajas que le han llevado a ser de los modelos más exitosos en este ámbito.

1.2. Objetivos

El objetivo principal es el de realizar un análisis del rendimiento de distintas configuraciones de parámetros sobre una red neuronal profunda que sirve como extractor de representaciones de locuciones independientemente de su duración y que constituye uno de los sistemas en el estado del arte en reconocimiento de locutor en los últimos años. Se hará un barrido de distintas dimensiones de red (distinto número de capas y distinto número de neuronas por capa), distintos datos de entrada, distintas arquitecturas de red, distintas dimensiones de vectores representación, etc. para al final dar una intuición de cuáles son las configuraciones óptimas y cómo influyen en el rendimiento final del sistema.

Por otra parte, se buscará analizar los efectos que tienen sobre el rendimiento técnicas de análisis factorial sobre estas representaciones de locuciones. Se estudiará el efecto de la técnica *"Linear Discriminant Analysis"* (LDA) y de la técnica *"Gaussian Probabilistic Linear Discriminant Analysis"* (GPLDA), y se tratará de ver bajo qué condiciones muestran mayor margen de mejora.

Como objetivo secundario, se evaluará el avance que pueda suponer este modelo con respecto al estado del arte actual, definido en el capítulo 2, y se estudiarán las ventajas que puede aportar con respecto a los sistemas de referencia.

1.3. Organización de la memoria

La memoria consta de las siguientes secciones:

1. **Introducción:** Motivación para la realización de este trabajo y sus objetivos.
2. **Estado del arte:** Presentación de los fundamentos del reconocimiento de locutor, breve explicación de los métodos empleados hasta la fecha en esta tarea e introducción de las tecnologías relacionadas con este trabajo.
3. **Entorno experimental:** Descripción de las herramientas y de las bases de datos empleadas.
4. **Diseño y desarrollo:** Planteamiento del trabajo realizado así como de la metodología seguida a lo largo del mismo. Se desglosa en etapas y fases detallando las medidas aplicadas en cada una de ellas, así como su razón de ser.
5. **Experimentos y resultados:** Exposición de los resultados obtenidos en los distintos experimentos realizados y comparativa entre ellos sobre distintos escenarios.
6. **Conclusiones y trabajo futuro:** Conclusiones finales obtenidas a partir de los resultados del apartado anterior y propuestas de guías de trabajo futuro en función de dichas conclusiones. Se incluye un desglose de las aportaciones externas a este trabajo.

2
Estado del arte

2.1. Sistemas tradicionales de reconocimiento de locutor

El reconocimiento de locutor se define como la tarea de extraer, caracterizar y reconocer la información de la locución que corresponde a la identidad de su autor[5] y engloba dos tareas más fundamentales: la identificación de locutor y la verificación de locutor.

La identificación de locutor consiste en determinar quién habla a partir de un conjunto de locutores conocidos, esto es, se efectúa una clasificación de tipo 1:N (referida comúnmente como identificación de conjunto cerrado) y no se realiza un reclamo de identidad por parte del autor del audio. La verificación de locutor, por otro lado, es la tarea de determinar si una persona es quien dice ser. Cabe destacar que en este caso se trata de una decisión de sí o no y debido a que los impostores (aquellos que reclaman una identidad que no es la correcta) no son conocidos por el sistema, nos encontramos ante una tarea de conjunto abierto.

El reconocimiento de locutor lleva consigo varios desafíos ligados que se han tratado de solventar con diferentes técnicas a lo largo del tiempo. Como se explica en [1], a diferencia de otras formas de biometría la voz humana es una biometría de rendimiento, lo que quiere decir que la información del locutor va incrustada en cómo se pronuncia la locución, no necesariamente en qué es lo que se está diciendo. Debido a esto, tendremos un alto grado de variabilidad entre una grabación y otra aún viniendo de la misma persona (lo que se conoce como variabilidad *"intra-speaker"*), ya que muchos factores modifican las características de un audio (la salud del autor, su edad, su estado anímico, el entorno de grabación, el dispositivo de grabación, etc.).

Dentro del reconocimiento de locutor y en función del grado de cooperación de los participantes podemos dividir los sistemas de reconocimiento automático entre dependientes del texto e independientes del texto [6]. En los sistemas dependientes del texto las frases a reconocer son fijas y conocidas, por ejemplo se puede pedir a los participantes que lean frases de un texto o una secuencia de números aleatorios como se describe en [7]. Mientras, en los sistemas independientes del texto no hay restricciones ni conocimiento de las palabras que el locutor emplea, como por ejemplo fragmentos de llamadas telefónicas o de conversaciones. Esto hace que la locución de entrenamiento y la locución de test puedan tener un contexto y una duración completamente diferente. Esta característica lo convierte en una tarea más compleja y desafiante, ya que, como explica [6], estos cambios en el contexto (tanto los debidos al locutor como los externos a él) aumentan la variabilidad *"intra-speaker"* dificultando el reconocimiento. Este proyecto se cen-

trará en los sistemas independientes del texto por su complejidad, por el elevado y creciente número de aplicaciones que tiene y por qué sigue suponiendo un desafío para el que se buscan continuamente técnicas que puedan perfeccionar el entorno actual.

Por otra parte, un problema compartido con otras disciplinas como el reconocimiento de idioma o de voz es que la señal de audio es una señal compleja que contiene mucha información, una parte de ella es útil para la diferenciación de locutor mientras que otra gran parte debe ser ignorada por los sistemas automáticos. Además, la variación de duración de las secuencias temporales que constituyen los fragmentos de audio supone un obstáculo en el modelado de esta información y su representación. En definitiva, hay un gran número de factores que hacen complicada la extracción de las partes útiles para el reconocimiento del resto de información no deseada, y las transformaciones complejas realizadas por determinados algoritmos de Machine Learning son ideales para modelar este tipo de datos[8].

Por todo ello, y motivado por el gran número de aplicaciones que tiene y tendrá esta disciplina, en los últimos años se han empleado diversas técnicas con el objetivo de encontrar modelos que sean capaces de obtener una representación de la señal que extraiga de ésta la información relacionada con el locutor e ignore toda la demás, reduciendo así la variabilidad *"intra-speaker"* y aumentando la variabilidad *"inter-speaker"*.

2.1.1. Modelos GMM-UBM

En torno a la década de los noventa y comienzo de los 2000 los *"Gaussian Mixture Models"* (GMM) se convirtieron en la principal técnica para el modelado de la señal (representada mediante una secuencia de vectores de características) en aplicaciones de reconocimiento de locutor independiente del texto [9]. Un GMM es una combinación de funciones de densidad de probabilidad que se emplea para modelar datos multivariados. Si adaptamos un GMM a las características de un locutor (generalmente características cepstrales a corto plazo[1]) podremos obtener una función de densidad de probabilidad asociada a dicho locutor que es la suma ponderada de las probabilidades de las distribuciones Gaussianas que la forman y si la evaluamos en un punto (por ejemplo con unas características obtenidas de un fragmento de audio de test) tenemos como resultado una medida de probabilidad que puede ser empleada para conocer la similitud entre la persona "referencia" con la persona autora del fragmento de test.

Aplicado a un entorno real de reconocimiento de locutor [10], se entrenan tantos GMM como locutores haya, obteniendo la función de densidad de probabilidad correspondiente a cada locutor. Posteriormente, el fragmento se evalúa con todas y cada una de las funciones y aquella que de una medida de probabilidad mayor corresponde al locutor cuya autoría es más probable.

Como se puede ver, obtenemos el locutor más similar entre un conjunto conocido (problema de clasificación 1:N) lo que nos resuelve el problema de la identificación de locutor. Sin embargo, para poder resolver el problema de la verificación de locutor se necesita aparte de los modelos para cada persona un modelo alternativo que represente al resto de locutores que no sean el objetivo. De esta manera podemos comparar ambos modelos (el del locutor objetivo y el alternativo) para obtener una decisión de aceptación o rechazo.

Este modelo alternativo, conocido como *"Universal Background Model"* (UBM) consiste esencialmente en un gran GMM entrenado para representar la distribución independiente de locutor de todas las personas disponibles y que se emplea como modelo alternativo para comparar con el GMM específico de cualquier persona objetivo[11]. Posteriormente, el UBM se comenzó a usar como modelo inicial a partir del cual se extraían los GMM específicos[9], en lugar de entrenar

[1] Las características cepstrales a corto plazo son características basadas en la frecuencia extraídas de fragmentos cortos de la locución (en torno a 20-25 milisegundos), como por ejemplo los *"Mel-frequency cepstral coefficients"* (MFCCs)

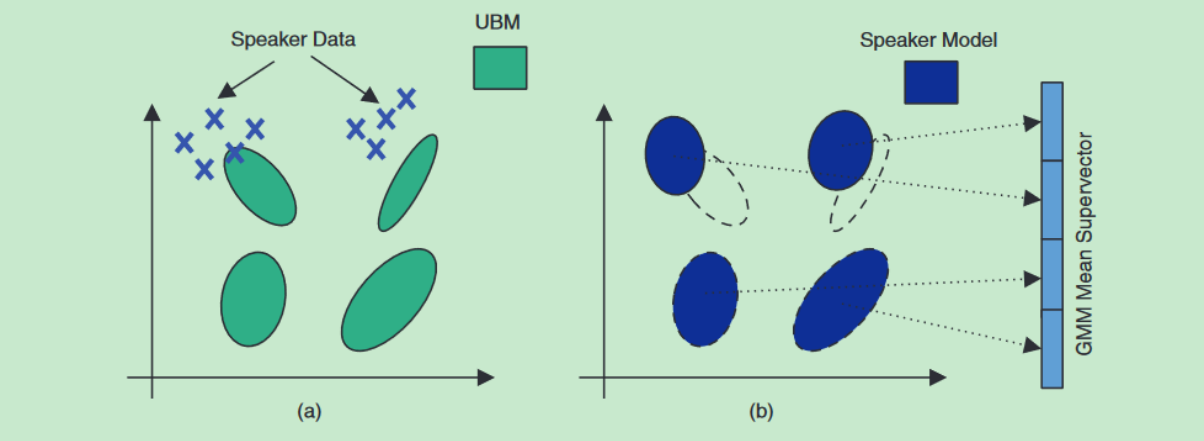


Figura 2.1: Entrenamiento de un GMM específico de locutor a partir de un modelo universal de todos los locutores registrados. Figura extraída de [1]

mediante la maximización de verosimilitud (ML) un GMM para una persona desde cero, se actualizan los parámetros de un modelo entrenado con suficientes datos como es el UBM mediante el método de la maximización a posteriori (MAP)[12], ver Fig, 2.1. Así, cada GMM específico contiene información de un locutor en concreto y también del resto de locutores empleados para la generación del UBM, lo que le otorga una mayor robustez ante la falta de datos de un locutor en concreto y permite obtener un mayor rendimiento que con GMM entrenados de manera independiente.

El modelo de cada locutor queda por tanto representado en un "supervector" que contiene las medias de las componentes Gaussianas que lo componen, formando una representación de longitud fija. Este vector tiene una dimensionalidad significativamente grande, se suelen emplear del orden de 1024 o 2048 componentes Gaussianas sobre 20 coeficientes MFCC aumentados con los coeficientes delta y doble delta[2], lo que compone un espacio de características de 60 dimensiones. Por tanto, un vector que represente un modelo contiene más de 60k dimensiones, lo que supone un desafío para el tratamiento o la comparación posterior entre modelos.

2.1.2. Modelos de Variabilidad Total: i-vector

Como se indica en el apartado anterior, los "supervectores" resultantes de representar el modelo universal y cada uno de los modelos específicos de cada locutor tienen una dimensionalidad significativamente grande. Es, por tanto, conveniente reducir la dimensionalidad de los vectores de representación de locutor y obtener únicamente la información útil para diferenciarlos. En este sentido cobra especial importancia la técnica del análisis factorial (FA) que busca dividir el espacio de características en dos subespacios diferenciados, uno relacionado con la variabilidad dependiente del canal y otro relacionado con la variabilidad dependiente del locutor, para finalmente describir la variabilidad de datos observables en una alta dimensionalidad empleando un número menor de variables "no observables".

Sin embargo, el análisis factorial tiene como contrapartida el hecho de que es necesaria una gran cantidad de datos para reunir la mayor variabilidad posible.

Como alternativa a la técnica FA y derivadas (como la JFA[3] [1]) surgen los modelos de varia-

[2]Los coeficientes delta y doble delta son parámetros que representan las propiedades dinámicas de los MFCC u otros coeficientes a corto plazo, y se calculan computando la velocidad y aceleración a lo largo de múltiples segmentos de la locución.

[3]La técnica JFA o joint factor analysis añade un subespacio adicional que contiene la información dependiente

bilidad total. Al observar que los factores de canal también contienen información dependiente del locutor, se unen los subespacios relacionados con la variabilidad de canal y de locutor en un único subespacio que contiene todos los tipos de variabilidad[13] (el espacio de variabilidad total). La creación de un único subespacio, que además tiene una baja dimensión, permite reducir el coste computacional del análisis factorial ya que se necesitan menos datos de entrenamiento para obtener una buena estimación de la variabilidad y además elimina los problemas derivados de trabajar sobre vectores de alta dimensión (como los generados por las técnicas GMM-UBM).

Más concretamente, la idea de esta aproximación es la de proyectar el supervector que representa un fragmento de habla al subespacio que se busca generar a través de una matriz de proyección previamente entrenada. Así, este modelo se representa de la siguiente manera:

$$\mu_{s,h} = \mu_0 + Tw_{s,h} \tag{2.1}$$

Donde $\mu_{s,h}$ es el supervector dependiente del locutor y del canal, μ_0 es el supervector proveniente del modelo universal y $w_{s,h}$ son las variables ocultas (no observables) fruto de la proyección y cuya estimación puntual con MAP nos da una representación de menor dimensión que el supervector original y que contiene información de la identidad del locutor en conjunto con otras fuentes de variabilidad (como la variabilidad de canal). Esta representación recibe el nombre de i-vector y ha sido la técnica predominante en el estado del arte en las últimas décadas.

2.2. Redes Neuronales Profundas

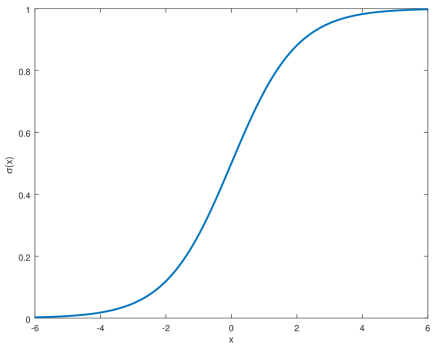
Una de las razones que hace que el rendimiento del cerebro humano y de otros algoritmos de aprendizaje sea tan diferente es la forma de extraer información útil de los datos. Se cree que el cerebro realiza una extracción de características y que, gradualmente, obtiene información de los datos a diferentes niveles de abstracción, obteniendo así distintos niveles de representación al descomponer el problema en una serie de sub-problemas de menor complejidad.

Este comportamiento es el que el paradigma de las redes neuronales quiere imitar, y por ello debe replicar algunas de las características presentes en el cerebro humano que le permiten realizar este tipo de transformaciones[14]. Estas características son: el paralelismo masivo, la computación y representación distribuida, la capacidad para generalizar, la adaptabilidad, el procesamiento de información contextual y la tolerancia a fallos.

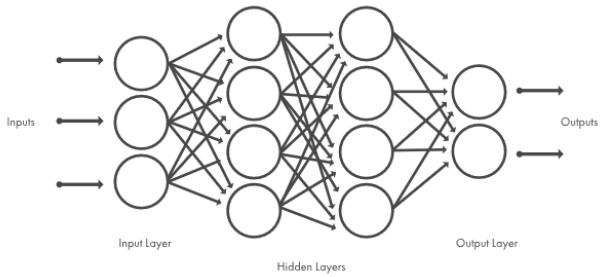
Se han realizado diversas aproximaciones a modelos que buscan cumplir las características antes mencionadas alejándose de la arquitectura tradicional de von Neumann de la computación moderna. Así, nos encontramos desde el modelo de Hebb o las redes de McCulloch-Pitts hasta ya modelos más actuales como las redes recurrentes, las redes convolucionales o las redes generativas antagónicas (GAN).

Estos modelos son una serie de algoritmos de aprendizaje automático de tipo distribuido. Se basan en un conjunto de elementos denominados neuronas que constituyen la unidad mínima de computación[15]. Estas neuronas se organizan en capas (de entrada, de salida y ocultas) comunicadas entre sí mediante conexiones ponderadas y su tarea es aplicar a la suma ponderada de las salidas de la capa anterior (la suma de la multiplicación de cada una de las salidas por los pesos de las conexiones con la neurona en cuestión) una función de activación, que es una función matemática generalmente no lineal que marca la salida de dicha neurona. Un ejemplo de función de activación es la función sigmoide, mostrada en la figura 2.2(a), aunque también se puede emplear la tangente hiperbólica, la tangente invertida, la función ReLU...

del locutor. De esta forma, al hacer una distinción entre la información dependiente e independiente del locutor, además del subespacio con la variabilidad de canal, se puede eliminar aún más información no relevante para el reconocimiento.



(a) Sigmoide binaria



(b) Ejemplo de red neuronal profunda

Figura 2.2: Ejemplos en las redes neuronales profundas

Además, el aprendizaje en este tipo de modelos se lleva a cabo a través de un algoritmo que actualiza los pesos de todas las conexiones con el objetivo de minimizar una función de coste (o de error), típicamente a través del descenso por gradiente estocástico. Por tanto, el aprendizaje no consiste en modificar el comportamiento de sus componentes (las neuronas) sino reforzar o debilitar la influencia que tienen según se va avanzando por la red. De esta forma, se consigue la descomposición del problema general en sub-problemas de menor complejidad a los que se va otorgando mayor o menor importancia en representaciones superiores. Como ejemplo, podríamos poner el reconocimiento de un dígito escrito a mano en una imagen, una primera capa podría ser la encargada (a partir de una entrada de bajo nivel como pueden ser los píxeles de la imagen) de reconocer formas muy simples, como esquinas, aristas o bordes. Una segunda capa, ya recibiendo dicha representación o descomposición inicial en lugar de píxeles, podría identificar formas más complejas como círculos o trazados compuestos y finalmente la tercera a partir de dichos trazados podría ser capaz de reconocer el dígito de la imagen. El algoritmo de aprendizaje sería el encargado de reforzar las conexiones de aquellas neuronas que son importantes en representaciones superiores (como dos líneas perpendiculares para formar una cruz) y debilitar las que son irrelevantes.

Muchas estructuras que se han empleado a lo largo de estos años no tienen la capacidad suficiente para aplicar las funciones complejas necesarias para realizar las transformaciones a los datos previamente mencionadas y así poder resolver ciertos problemas. Las arquitecturas clásicas disponen de una única capa oculta[4] lo que implica una única transformación no lineal. Es aquí donde surge la idea de las redes neuronales profundas (*"Deep Neural Networks"*, DNNs), que son aquellas que cuentan con dos o más capas ocultas. El hecho de incluir varias capas ocultas les permite extraer características con múltiples niveles de abstracción y, por tanto, aprender así relaciones no lineales de mayor complejidad (si se requieren más transformaciones, se apilarán más capas). Un ejemplo gráfico de una red neuronal profunda con dos capas ocultas se puede ver en la figura 2.2(b).

Sin embargo, el uso de arquitecturas profundas trae consigo algunas desventajas. Cuanto más compleja es la arquitectura empleada (y, por tanto, la función que modela) mayor número de mínimos locales habrá, lo que puede dificultar el aprendizaje. Esto ha causado que durante mucho tiempo algoritmos de aprendizaje como *"Back-propagation"* o el descenso por gradiente no hayan obtenido buenos resultados en los experimentos con estas nuevas arquitecturas. Por otra parte, al aumentar la complejidad de la red aumenta el número de parámetros que deben ser aprendidos, por lo que la capacidad de cómputo de los ordenadores debe ser mayor y el

[4]Una capa oculta es cualquier capa dentro de una red neuronal que se encuentra tras la capa de entrada o de recepción de los datos y antes de la capa de salida o de extracción del resultado.

número de datos empleados en el entrenamiento también debe aumentar. Además existen algunos problemas abiertos, como la selección de los hiper-parámetros y la estructura (topología) de la red de acuerdo a una determinada tarea.

El interés en las redes neuronales ha ido aumentando en los últimos años, y actualmente estos métodos forman parte del estado del arte en muchos de los ámbitos de la inteligencia artificial. Esto es debido a la ya mencionada capacidad para aprender representaciones complejas no lineales de la entrada, al hecho de que hasta ahora no se disponían de grandes bases de datos que solventaran las carencias que causaban bajo rendimiento en los algoritmos de entrenamiento, al incremento de la potencia computacional así como los avances en los sistemas basados en procesamiento gráfico (que ahora permiten procesar más datos en un período menor de tiempo) y a una evolución de las técnicas empleadas que ha permitido ir solventando algunas de las desventajas y limitaciones que tenían este tipo de técnicas. Por ejemplo, hasta 2006 no se consiguió tener éxito en arquitecturas profundas con más de dos capas ocultas (salvo para redes convolucionales) debido a limitaciones en el entrenamiento de este tipo de estructuras[16], más concretamente debido a los datos empleados y a la inicialización de los pesos seleccionada.

Actualmente existe una gran cantidad de campos en los que las redes neuronales son especialmente interesantes por sus características ya mencionadas, como el reconocimiento de patrones (imágenes, caracteres...), predicción y clasificación de series temporales, optimización, detección de fraude, etc. Y su uso está cada vez más extendido tanto en el ámbito industrial como en el ámbito de la investigación.

2.3. Redes neuronales aplicadas al reconocimiento de locutor

Como ya se ha mencionado anteriormente, las señales de audio son señales complejas que contienen mucha información, una parte útil y otra que no aporta nada en relación a la tarea en cuestión. Es por esto que las redes neuronales son, a priori, una técnica muy interesante en este ámbito, por su capacidad de transformar las características para obtener niveles de representación diferentes y aprender cuáles son aquellos factores que son más útiles en lo relacionado con el reconocimiento de locutor. Además, en los últimos años los datos etiquetados[5] han aumentado considerablemente tanto en tamaño como en calidad, en gran parte debido al trabajo de organizaciones como NIST (*"National Institute of Standards and Technology"*) que organizan anualmente series como las *"Speaker Recognition Evaluations"* (SREs) y que proporcionan un conjunto de datos y un entorno de trabajo estandarizado para todos los participantes. Este aumento en la cantidad de datos disponibles ha hecho que las redes neuronales cobren especial importancia en este tipo de evaluaciones.

Por otro lado, numerosos casos de éxito en tareas similares (como el reconocimiento de idioma o de habla) dan una intuición de la idoneidad de las redes neuronales en estos ámbitos. Una de las primeras aplicaciones exitosas[17] reemplazaba los GMMs por DNNs para el reconocimiento de habla, lo que mejoraba en 9 puntos el *"Word error rate"* obtenido por los GMMs. Tras esto, muchos otros experimentos mostraron un sorprendente rendimiento en esta misma tarea mejorando cada uno de ellos el resultado anterior, lo que lo coloca como una línea de investigación predominante por el margen de mejora que tiene. Ejemplo de ello es [18], que emplea redes convolucionales (CNNs) para mejorar el híbrido DNN-Hidden Markov Model de [17], o [19] que consigue un mejor rendimiento empleando redes neuronales recurrentes.

También cabe destacar la similitud entre el reconocimiento de locutor y reconocimiento de idioma, donde las técnicas que muestran mejores resultados son trasladadas de una a otra

[5]Los datos etiquetados son aquellos datos que tienen información de la clase a la que pertenecen. Son necesarios para poder realizar técnicas de aprendizaje supervisado, en el que se compara la predicción con la clase objetivo.

disciplina con éxito. Un ejemplo de ello se muestra en [20] donde se presenta una estructura de red neuronal que mejora los resultados obtenidos por la técnica GMM-UBM tanto en el reconocimiento de locutor como en el de idioma. Este estudio presenta una idea que forma parte de muchas de las aproximaciones basadas en DNNs y es el reemplazo de los GMMs como técnica de modelado de los locutores (o del idioma). Lo que se hace es sustituir los resultados obtenidos por un modelo tradicional de GMM-UBM por los resultados de una red neuronal entrenada de manera específica para estas tareas y, en algunos casos, se emplean esos resultados para generar nuevos vectores de características. Quizás el ejemplo más utilizado de vector de características extraído de una red neuronal son las características de cuello de botella (*bottleneck features*)[21], que han reemplazado gradualmente a otras características acústicas como los MFCCs o los coeficientes de predicción lineal perceptual (PLP).

2.3.1. DNN-Embeddings como representaciones a nivel de locución

Uno de los puntos débiles de los vectores de características como los MFCC, los coeficientes PLP o las características de cuello de botella son que se extraen por cada trama (*frame*) de una locución, lo que genera una secuencia dependiente de la longitud de la locución y esto supone un desafío para el modelado posterior. Es, por tanto, muy interesante obtener una representación de longitud fija de la locución completa, de manera similar a los i-vectors. En este trabajo se emplea la representación conocida como *"embedding"* que consiste en dividir la red en dos partes, una parte a nivel de trama y otra parte a nivel de locución (o secuencia). La primera parte realiza las transformaciones de los vectores de características (como pueden ser los MFCC u otras características a nivel de trama) y la salida de la última capa de esta parte se pasa como entrada a una capa de estimación de estadísticos (resumen de información), que calcula la media y la desviación típica para la salida correspondiente a cada trama y así forma un único vector representando la locución completa. Este vector se convierte en la entrada de la parte de la red que trabaja directamente sobre representaciones de locuciones. Todo ello se optimiza como un único modelo.

Por tanto, si entrenamos una red para el reconocimiento de locutor, en las capas que constituyen la parte "a nivel de locución" de la red estaremos obteniendo representaciones de las locuciones que buscan contener toda la información relativa al locutor (con mayor o menor nivel de abstracción en función de la capa que se seleccione, más o menos cerca a la salida). De esta forma, estamos obteniendo un vector que representa a la locución completa y que está enfocado a la tarea específica que tratamos de abordar (nótese la similitud con las técnicas de análisis factorial y los subespacios relacionados con la variabilidad de canal y de locutor).

Trabajos como [3] en el que se usa una arquitectura con retraso de tiempo (*"Time Delay Neural Network"*, TDNN) para la verificación de locutor o [22] en el que se emplea una arquitectura *"Long Short-Term Memory"* (LSTM) bidireccional para el reconocimiento de idioma son buenos ejemplos de la aplicación con éxito de la técnica de embeddings como representaciones de locuciones.

3

Entorno experimental

En esta sección de la memoria se detallarán las herramientas y las bases de datos empleadas a lo largo de todo el desarrollo del trabajo. Cabe destacar que estas herramientas son las utilizadas en la actualidad tanto en investigación como en industria.

3.1. Herramientas empleadas

3.1.1. Kaldi

Kaldi [23] es una herramienta de código abierto que ofrece funcionalidad para el procesamiento de voz con el objetivo de ser empleada en el ámbito de la investigación al tener un código moderno y flexible, fácil de modificar y ajustar a cada tarea. Está escrita en C++ bajo la licencia "Apache License v2.0" y se diferencia de otros *"toolkits"* en la integración de transductores de estado finitos, un extenso soporte para álgebra lineal y un diseño genérico y extensible.

Aunque su uso queda fuera del ámbito de este Trabajo de Fin de Grado, ya que no se ha empleado en los experimentos realizados, sí ha sido necesario su uso en la obtención de los coeficientes cepstrales en las frecuencias de Mel (MFCCs) a partir de los audios disponibles en las distintas bases de datos.

3.1.2. TensorFlow

TensorFlow [24] es una librería de código abierto que emplea gráficos de flujo de datos para la computación numérica. Desarrollada por Google, su principal objetivo es el de servir de plataforma *end-to-end* con distintos niveles de abstracción para simplificar el desarrollo de sistemas de aprendizaje automático (aunque es lo suficientemente general como para ser aplicable a otros ámbitos). Se basa en grafos, que son una serie de operaciones y datos enlazados, de forma que un gráfico define el comportamiento del entorno al definir cuáles son los datos y cómo se interrelacionan entre sí a través de las operaciones. La unidad central de datos en tensorflow se denomina *tensor*, que consiste en un conjunto de valores primitivos colocados dentro de una matriz de cualquier número de dimensiones. Se trata de un entorno multi-plataforma que tiene integración con procesamiento gráfico y paralelo, por lo que una de sus principales ventajas es

que nos permite un uso transparente de las GPUs agilizando los cálculos hasta un factor de 140 con respecto al cálculo en CPU. En definitiva, encapsula el manejo de datos y operaciones entre dispositivos de forma que se optimice el rendimiento de una forma transparente para el usuario.

Gracias al uso de tensores y grafos, tensorflow permite definir y evaluar expresiones matemáticas con matrices multidimensionales de una manera muy eficiente tanto en CPU como en GPU, algo que es realmente útil en el ámbito del aprendizaje automático y más concretamente en las redes neuronales por la cantidad de operaciones de este estilo que es necesario computar.

Además, el empleo de grafos y sesiones le otorga una modularidad que le da la capacidad de ser integrado en diferentes entornos y plataformas de una forma simple. Y su alto nivel de integración con otras tecnologías (API funcional de Keras, BERT, Ragged Tensors...) permite añadir nueva funcionalidad o aumentar la ya existente de una forma rápida y sencilla.

En este trabajo se empleará Tensorflow como plataforma para la implementación, entrenamiento y evaluación de las redes neuronales profundas, así como la extracción de las representaciones de locuciones (embeddings).

3.1.3. Keras

Keras [25] es una API de redes neuronales de alto nivel para las plataformas de Theano, TensorFlow y CNTK. Está escrita en Python y se ha desarrollado con el objetivo de simplificar y agilizar la implementación y experimentación de redes neuronales de distinto tipo sin reducir la flexibilidad (según sus autores: *"Being able to go from idea to result with the least possible delay is the key to doing good research"*). Se basa en los principios del desarrollo centrado en el usuario, la modularidad y la facilidad de extensión y busca ser un envoltorio único para diferentes plataformas que conforman su núcleo.

La principal ventaja de esta API es que encapsula todo el proceso de desarrollo y explotación de una red neuronal eliminando toda la implementación a desarrollar si se usara directamente el núcleo que envuelve. Esto nos da la ventaja de que no necesitamos gestionar por ejemplo tensores si estamos empleando TensorFlow y sobre todo que no tenemos que modificar toda la implementación si decidimos cambiar por ejemplo a las matrices de Theano, ya que la interfaz de Keras para ambos núcleos es la misma e internamente gestiona las estructuras necesarias. Esto nos permite agilizar los cambios en la estructura y hacer más eficiente el proceso de barrido de arquitecturas y el cambio de parámetros en los experimentos.

Además, Keras comparte las ventajas de TensorFlow y Theano en cuanto al soporte de paralelismo en múltiples CPUs o GPUs y al emplear estas plataformas por debajo realiza los cálculos con matrices multidimensionales de una manera muy eficiente.

En este trabajo se emplea Keras como API para tensorflow, gracias a ella se realiza un análisis y comparativa de las distintas arquitecturas y tipos de las redes neuronales estudiadas en ese TFG, así como un barrido de los parámetros que modifican el comportamiento de éstas.

3.1.4. Matlab

Matlab es un sistema de cómputo numérico propietario con su propio lenguaje de programación desarrollado por MathWorks. Entre sus prestaciones podemos encontrar la manipulación de matrices, la representación de datos y funciones y la implementación muy eficiente de algoritmos. Por su versatilidad en muchos campos de las matemáticas y la ingeniería se ha convertido en uno de los sistemas más usados en universidades y centros de investigación. Es un software que dispone de una gran cantidad de *"Toolkits"* destinados a aumentar su funcionalidad, y cada año se aumenta el catálogo de complementos que recibe esta plataforma.

En este trabajo aprovecharemos las capacidades de Matlab en el manejo eficiente de matrices para aplicar estrategias de compensación (LDA y GPLDA) a los vectores de representación de las locuciones (i-vectors y embeddings). Finalmente, también emplearemos Matlab para evaluar el rendimiento del sistema de reconocimiento de locutor analizado.

3.2. Bases de datos

Para el desarrollo de este trabajo se han empleado datos provenientes de cuatro bases de datos diferentes para entrenamiento, y los datos de una quinta para el test. No obstante, se ha aplicado un tratamiento previo de éstos ya que nuestra red no trabaja sobre el audio en bruto, sino sobre conjuntos de coeficientes extraídos a partir de estos audios. Una vez se ha aplicado el preprocesamiento se juntan las primeras cuatro bases de datos en un gran conjunto unificado de entrenamiento.

3.2.1. NIST datasets

En este trabajo se ha empleado para entrenamiento una sección de las recopilaciones de datos empleadas en las evaluaciones de reconocimiento de locutor de 2004, 2005, 2006 y 2008 y para test una sección de la evaluación del 2010. Todas estas evaluaciones están organizadas por el *National Institute of Standards and Technology, (NIST).* Este organismo anual o bi-anualmente realiza una serie de evaluaciones en distintos campos del procesamiento de audio con el objetivo de explorar nuevas ideas y comparar las tecnologías desarrolladas en el mismo entorno para el mismo problema, y para ello provee de una base de datos centrada en la tarea, extensa, estandarizada y bien etiquetada (con un porcentaje muy bajo de errores en las etiquetas).

Estas bases de datos contienen fragmentos de audio grabados por el *Linguistic Data Consortium* (LDC) a partir de un gran conjunto de locutores de manera simultánea empleando numerosos micrófonos en situaciones comunicativas diferentes y en distintos idiomas. El idioma de los audios es principalmente inglés aunque también se incluyen locuciones en árabe, bengalí, chino, persa, hindi, coreano, ruso, español, tailandés y urdu. Los audios son segmentos multi-canal de conversaciones telefónicas grabados simultáneamente desde diferentes micrófonos auxiliares y de una longitud variable, con un formato de almacenaje de 8 bits en ficheros SPHERE, con información adicional como por ejemplo el idioma empleado.

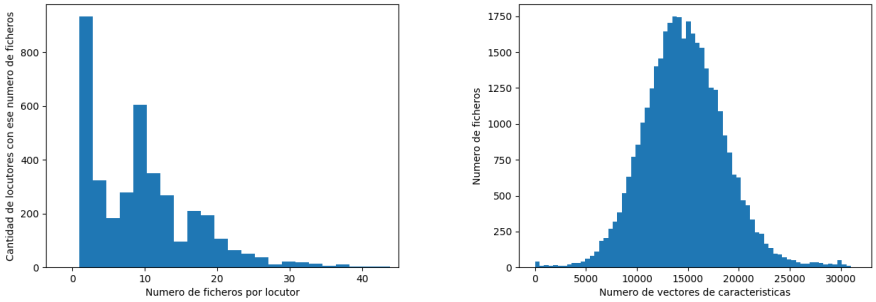
3.2.2. Coeficientes cepstrales en las frecuencias de Mel (MFCC)

Todos los ficheros de audio son tratados empleando Kaldi para obtener una matriz con los coeficientes cepstrales en las frecuencias de Mel (MFCCs), que sustituirán al audio en bruto como características empleadas para entrenar la red extractora de embeddings y luego se utilizarán para extraer las representaciones de las locuciones.

Los MFCCs son coeficientes que representan el habla basados en la percepción auditiva humana. Son características a corto plazo, por lo que no podrán ser usadas para representar la locución completa, sino segmentos de pequeña duración (típicamente 20 o 25 milisegundos). Los MFCCs han demostrado ser características muy útiles en el modelado de locuciones para el reconocimiento de locutor y de idioma, mostrando una mejora de rendimiento con respecto a otras técnicas [26].

Base de datos	Nº de locuciones
SRE 2004	4528
SRE 2005	2735
SRE 2006	18185
SRE 2008	10973
Total	36421

Tabla 3.1: Reparto de locuciones por base de datos en el conjunto final



(a) Distribución de la cantidad de ficheros por locutor

(b) Distribución de la longitud de las locuciones

Figura 3.1: Ejemplos en las redes neuronales profundas

3.2.3. Análisis del conjunto de datos

En total contamos con 36421 ficheros que corresponden cada uno de ellos a una locución diferente (y, por tanto, cada uno de ellos contiene una secuencia de vectores de MFCCs). La distribución de los ficheros por las distintas bases de datos queda como se indica en la tabla 3.1. En total, estos ficheros pertenecen a 3794 locutores diferentes pero por la diferencia en número de locuciones por cada uno de ellos hemos decidido aplicar distintos filtros (como se detalla en la sección 4.1). Además, contamos con el problema de la gran diferencia en la longitud entre los distintos audios, por lo que habrá que aplicar algún tipo de medida para gestionar esta dificultad. Un análisis de la distribución de la longitud de las distintas locuciones y la distribución del número de ficheros por locutor se puede encontrar en la figura 3.1.

4

Diseño y Desarrollo

Todos los experimentos desarrollados en este trabajo de fin de grado pasan por tres etapas muy diferenciadas que se corresponden con los pasos lógicos en la creación de un sistema de reconocimiento de locutor basado en embeddings y su evaluación. Así, las tres etapas son:

1. Entrenamiento del extractor de embeddings
2. Extracción de embeddings
3. Evaluación de dichos embeddings

Además, todos ellos comparten una fase inicial que debe ser realizada una única vez al comienzo de la experimentación (la salida de esta fase es compartida por todos los experimentos realizados) que consiste en la parametrización de la base de datos (extracción de MFCCs) y la creación de ficheros auxiliares para facilitar el entrenamiento de la red neuronal que sirve como extractora de embeddings. Hay que destacar que aunque se ha mencionado que esta fase es común para todos los experimentos, hay algunos matices y diferencias en algunos de ellos. Estos matices quedarán explicados en la sección 4.1.

Aunque todos los experimentos deben pasar por las cuatro fases, explicaremos el diseño de cada fase de manera separada debido a la independencia entre ellas, ya que los parámetros o condiciones de una fase no influyen en fases previas o posteriores. Se puede pensar un experimento como un conjunto separado de sub-experimentos diferentes que producen resultados que sirven como entrada de la siguiente fase. De hecho, en varias ocasiones se toma la salida parcial de un experimento como entrada de otro, evitando tener que repetir las primeras etapas de la experimentación.

4.1. Parametrización y preprocesado de los datos

En esta primera fase se han creado scripts de bash y python para el manejo de ficheros de texto y ficheros con extensión *"csv"* y *"feat"*.

Aunque su extracción queda fuera del ámbito de este trabajo, es interesante mencionar el proceso de extracción de las características (MFCCs) que se ha llevado a cabo. Como se indica en

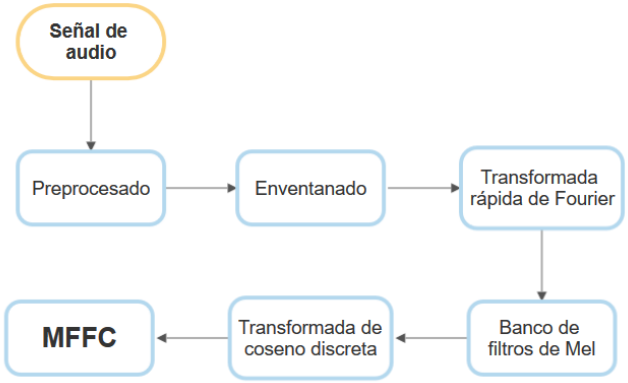


Figura 4.1: Proceso de extracción de los coeficientes cepstrales en las frecuencias de Mel (MFCC). Figura basada en [2].

el apartado 3.2.2, los MFCCs son características cepstrales a corto plazo, típicamente en ventanas de 20 a 25 milisegundos. Por tanto, el primer paso es el de extraer tramas de 20 milisegundos con un solapamiento de 10 milisegundos entre ellas, a las que se le aplica un enventanado Hamming[1]. Tras aplicar el enventanado, se realiza la transformada rápida de Fourier, que es un algoritmo eficiente que permite calcular la transformada discreta de Fourier[2], a la que se le aplica un banco de filtros de escala perceptual de Mel. Este banco de filtros produce la energía espectral en cada canal representando las diferentes bandas de frecuencia, y en nuestro caso contamos con 24 filtros con unas frecuencias desde 300 Hz hasta 3300 Hz. Una vez hecho esto, lo siguiente es calcular el logaritmo de cada una de estas energías espectrales y finalmente realizar una transformación directa del coseno (DCT), de la cual se obtienen los 20 coeficientes con más información. Este vector de 20 coeficientes es el vector de MFCCs correspondiente a una ventana de 20 milisegundos, por lo que si realizamos este proceso para todas las ventanas que conforman un audio obtendremos una matriz de MFCCs que serán las características propias de una locución. Este proceso se puede observar en la figura 4.1.

Existen una serie de razones por las que no es adecuado emplear los ficheros de características sin realizar ningún tipo de ajuste. Algunas de ellas se pueden deducir del análisis realizado en el apartado 3.2 y otras se observan al analizar los ficheros de características MFCC ya extraídos o los ficheros auxiliares como la lista de etiquetas[3]. Por tanto, será necesario un procesamiento previo para dejarlos preparados con el formato y la distribución que requiere la red que vamos a entrenar. Las razones que impiden que los datos puedan ser usados directamente son las siguientes:

- Existen locuciones no etiquetadas (sin información de locutor) y etiquetas sin locuciones correspondientes.
- La cantidad de locuciones por autor es muy diferente (datos desbalanceados).
- Las etiquetas no están en un formato adecuado para la red a entrenar.
- Los datos no están normalizados.

[1]Las ventanas son funciones matemáticas empleadas en el procesamiento de señales para eliminar las discontinuidades al comienzo y al final de las tramas analizadas.

[2]La transformada discreta de Fourier es un tipo de transformada usada en el análisis de Fourier que convierte una función matemática en el dominio del tiempo en otra función en el dominio de la frecuencia.

[3]La lista de etiquetas es una lista que une cada identificador de fichero de características con el identificador de su autor.

Para solventar todos estos problemas se ha desarrollado una serie de ficheros de código python y scrips de bash de tratamiento de los datos que, ejecutados en orden, van generando una serie de ficheros parciales que servirán como apoyo al entrenamiento de la red y que incluyen medidas para los distintos contratiempos de la base de datos. Las medidas aplicadas son las siguientes:

1. **Filtrado de los ficheros válidos:** Analizando la lista de etiquetas se pudo ver que existen ficheros de los que no se tiene información sobre su locutor y etiquetas que no tienen un fichero de características asociado. Al suceder esto en la base de datos dedicada al entrenamiento y a la validación de la red neuronal, y tratarse de un problema de aprendizaje supervisado, no podemos emplear locuciones sin información de su autor. Es necesario, por tanto, crear una nueva lista de etiquetas que contenga de la lista original sólo los ficheros que se disponen en la base de datos, de esta forma al coger únicamente los ficheros de esta nueva lista filtrada nos aseguramos que todos los ficheros seleccionados existen y tienen información sobre su autor.

   En la lista de etiquetas original contábamos con 36612 entradas, mientras que tras el filtrado se reduce a 36421, lo que quiere decir que 191 ficheros de los que se disponía la información de locutor no se encuentran en la base de datos. Por otro lado, contamos con 54693 ficheros, de lo que se deduce que de 18272 ficheros no se dispone de información de su autor y deben ser descartados.

2. **Selección de un número fijo de locuciones por locutor:** Si observamos la figura 3.1(a) podemos observar que el número de locuciones por locutor no es constante y tiene un alto grado de variabilidad. El uso de datos desbalanceados para el entrenamiento de redes neuronales puede causar que las características que representen a los locutores con pocos audios sean ignoradas[27] y se tienda a clasificar por defecto hacia aquellos con mayor número de ficheros, una tendencia que supone un peor entrenamiento y como consecuencia tiene un menor rendimiento.

   La medida para balancear los datos empleada es la fijación de un número determinado de locuciones por locutor. Siendo N el número fijo de locuciones, se descartan todos los locutores con menos de N locuciones, y se trunca a N locuciones a aquellos locutores que tienen más de N ficheros. Además, aprovechando que se tiene ya fijo el número de locuciones a N por persona, el reparto entre los conjuntos de entrenamiento y validación es simple, se puede seleccionar el porcentaje de datos empleados para validación simplemente seleccionando el porcentaje de N empleado para validación. Así, se obtienen dos conjuntos balanceados con datos de todos los locutores del conjunto original. Sin embargo, esta aproximación tiene el problema de que se están dejando datos sin utilizar.

   También se ha implementado la separación en entrenamiento y validación sin realizar un balanceado de los datos, con el objetivo de comparar el rendimiento de la red con balanceado y sin balanceado (y por consiguiente, con un número distinto de locutores y de datos). En este caso se sigue haciendo un descarte de los locutores con menos de un determinado número de locuciones pero ya no se trunca el número de ficheros que puede tener una persona.

   En este trabajo se ha fijado N = 13 si se balancean los datos, de los cuales 2 ficheros se toman como validación. Esto nos deja con un total de 1124 locutores y 14612 ficheros de características (12364 para entrenamiento y 2248 para validación). En el caso de no balancear los datos, se han eliminado todos los locutores con menos de 5 ficheros y de los restantes se han tomado todos sus ficheros menos dos, que se tratan como validación. En esta opción contamos con 2539 locutores, 29015 ficheros para entrenamiento y 5078 para validación.

Id. locutor	Id. numérico	Id. de locutor	Id. numérico
100304-f	0	8975-m	2534
100396-f	1	8976-f	2535
1004-f	2	8977-f	2536
1006-m	3	8988-f	2537
1008-f	4	8993-f	2538

Tabla 4.1: Relación entre los identificadores de los locutores y su correspondiente identificador numérico

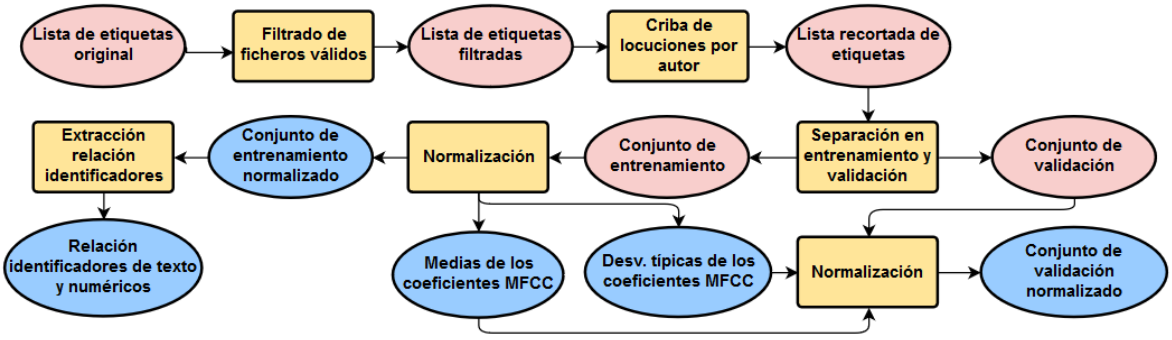


Figura 4.2: Flujo de preprocesado de los datos

3. **Modificación del formato de las etiquetas:** Como información de locutor contamos con etiquetas que son cadenas de texto, un formato incómodo para su manejo, e inadecuado cuando tratamos con redes neuronales. Será necesaria la creación de una relación de identificador de texto a un correspondiente identificador numérico, de forma que luego pueda ser transformado a categórico y empleado para el cálculo del coste durante el entrenamiento. Un ejemplo de relación para varios identificadores de texto en la versión no balanceada se puede encontrar en la tabla 4.1.

4. **Normalización de los datos:** La normalización en general facilita el aprendizaje de la red. Esto es debido principalmente a que valores muy lejanos del cero en las funciones de transferencia (como por ejemplo la sigmoide de la figura 2.2(a)) caen en regiones ya muy saturadas, por lo que su gradiente es prácticamente cero y la modificación de pesos es mínima. La normalización centra los datos en torno al cero con una varianza de la unidad para eliminar este riesgo, por lo que aplicaremos esta técnica a nuestras matrices de MFCCs para que cada uno de los 20 coeficientes esté normalizado (sobre el conjunto de entrenamiento).

Todas estas medidas deben ser aplicadas iterativamente, ya que la aplicación de una depende de la salida de la anterior. Así, se crea un flujo de datos que da como resultado final las matrices de entrenamiento y validación y una serie de ficheros auxiliares que no sirven sólo para el entrenamiento del extractor de embeddings sino también para fases posteriores, como la relación entre identificadores de locutores y su identificador numérico o las medias y las desviaciones típicas de cada uno de los 20 coeficientes MFCC. Se puede ver el flujo de tratamiento de los datos en la figura 4.2, donde los cuadros amarillos representan las operaciones sobre los datos, las elipses naranjas los resultados parciales y las elipses azules los ficheros de datos finales, que serán empleados en el resto del proceso.

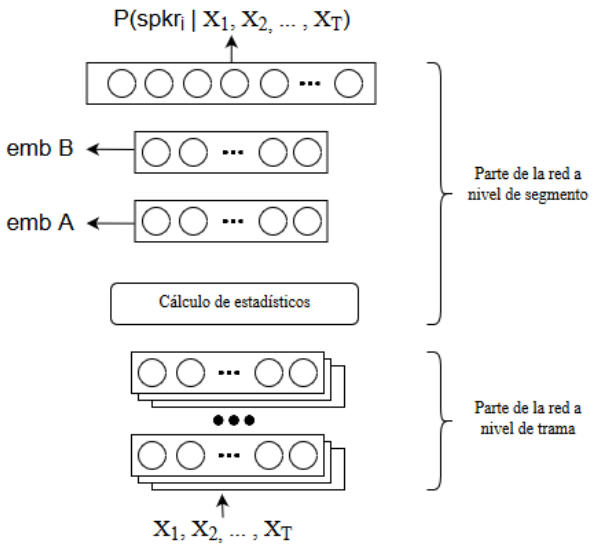


Figura 4.3: Estructura y componentes de la red neuronal extractora de embeddings. Figura basada en [3].

4.2. Entrenamiento del extractor de embeddings

En esta segunda fase se ha empleado la API funcional Keras junto con TensorFlow para la definición, entrenamiento y evaluación de los distintos extractores de características.

Una vez ya están las matrices de características preparadas el siguiente paso será entrenar el modelo que sirve como extractor de embeddings. Como ya se indicó en el apartado 2.3.1, los embeddings son representaciones de locuciones completas independientes de la longitud de éstas y con información específica de la tarea a realizar (en este caso la representación del autor de la locución). Cabe destacar sus diferencias con los vectores de MFCCs, que son útiles a corto plazo (por lo que al tener que enventanar no son independientes de la longitud de la locución) y no son específicos de la tarea a tratar, sino que contienen la información acústica de la señal.

Es necesario, por tanto, un sistema que sea capaz de extraer la información dependiente del locutor y por tanto relevante para su identificación y agruparla a lo largo del tiempo (de ventana en ventana), de forma que al final se obtenga un vector que sea el resumen de todas estas ventanas temporales y que contenga sólo la información necesaria para la tarea de reconocimiento de locutor.

En este sentido resulta especialmente interesante la red neuronal profunda como modelo extractor. Primero, por su capacidad para realizar transformaciones complejas no lineales y crear diferentes niveles de abstracción sobre los atributos de entrada, lo que le permite crear distintas representaciones de estos atributos en cada capa. Segundo, por su forma de aprendizaje, en la que otorga mayor o menor importancia a cada característica en función de la tarea que define su función de coste, y por tanto permite definir aquellas características que influyen en el resultado final y que le permiten tener éxito. En definitiva, por su forma de aprender, acaba discriminando entre aquellos atributos que le ayudan para una determinada tarea y aquellos que no, y dando un grado de importancia a cada uno de ellos. En este caso, al aplicarlo a la tarea de identificación de locutor, será capaz de crear distintas representaciones partiendo de los MFCCs y seleccionar los atributos que son más útiles a la hora de determinar quién es el autor del audio que representan los coeficientes. De esta forma conseguimos esta cualidad de los embeddings de ser representaciones específicas de la tarea a abordar.

En tercer lugar, una DNN resulta especialmente interesante por todos los escenarios en los que

ya ha demostrado ser una herramienta especialmente útil para la extracción de características, ya sea a nivel de ventana o a nivel de locución completa. Así, tenemos el ejemplo de las redes neuronales de tipo *"Bottleneck"*, donde se emplea una capa de menor dimensión para comprimir la información necesaria en un vector de características más pequeño [28] o, ya dentro del mismo ámbito que este trabajo, los modelos propuestos en [3] y en [22], que usan redes neuronales para extraer embeddings en verificación de locutor y reconocimiento de idioma, respectivamente, y que obtienen resultados que mejoran el resto de técnicas de representación.

Por ello, en este trabajo se empleará una estructura de red neuronal profunda consistente en dividir la red en dos partes separadas con una capa intermedia. La primera parte maneja los datos a nivel de ventana, recibe como entrada los distintos vectores de MFCCs y su objetivo es realizar una transformación de éstos para obtener otros niveles de representación. El resultado obtenido tras la salida es un nuevo vector de características por cada una de las ventanas (sigue siendo dependiente de la longitud del audio). Tras ello, los datos pasan por una capa de agrupación de estadísticos (*"Statistics pooling"*) cuyo objetivo es agrupar los valores de cada uno de los elementos del vector de salida de la parte anterior para cada una de las ventanas de la locución en un único vector que represente la locución completa. Para ello, calcula la media y la desviación típica a lo largo del tiempo (a lo largo de las diferentes ventanas) para todos los atributos de aquella salida y genera este único vector por locución que supone la entrada de la segunda parte de la red. En la figura 4.3, extraída de [3], podemos observar un diagrama genérico de las partes que componen la red y su arquitectura.

La segunda parte de la red ya trabaja a nivel de locución. Recibe vectores de estadísticos que resumen la información de las locuciones completas y realiza transformaciones sobre éstos para acabar clasificando quién es el autor más probable del audio correspondiente de entre un conjunto de autores conocidos (tarea de identificación de locutor).

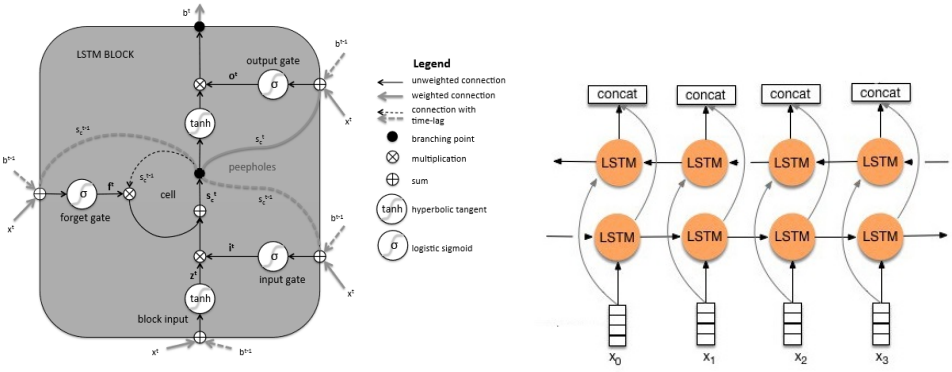
Se debe destacar la cantidad de parámetros a ajustar en un modelo como el descrito, por lo que parte de este trabajo de fin de grado se centra en realizar un análisis de la configuración de parámetros que maximiza el rendimiento de la red, así como de las distintas técnicas a aplicar para mejorar los resultados. Por las diferencias en la configuración y el funcionamiento de ambas partes de la red, analizaremos los parámetros y técnicas relacionados con cada una de ellas de manera separada.

4.2.1. Parte a nivel de ventana

Esta parte de la red recibe como datos de entrada una secuencia de vectores de 20 características MFCC. Es importante indicar que esta secuencia tiene una gran dependencia temporal, ya que lo que diferencia a un vector del siguiente es un período de tiempo bastante pequeño (10 milisegundos) y están temporalmente solapados, por lo que la información de un fragmento tiene mucho que ver con los fragmentos anteriores y posteriores. Por esta condición, se han empleado tipos de redes neuronales que buscan explotar el contexto y las dependencias temporales, y son una alternativa muy interesante para tratar este tipo de datos (ya que éstas dependencias dentro de la secuencia tienen información dependiente del locutor). Se han explorado dos arquitecturas diferentes inspiradas en modelos ya existentes por su alto rendimiento en tareas similares [3][22]: la arquitectura BLSTM y la arquitectura TDNN.

Bidirectional Long short-term memory

La arquitectura *"Long short-term memory" (LSTM)* es una arquitectura del tipo recurrente, lo que quiere decir que incluye retroalimentación de una neurona a sí misma. Esta retroalimentación le da la capacidad de tener una salida no dependiente únicamente de la entrada recibida,



(a) Operaciones en una neurona de tipo LSTM. Figura extraída de [29]

(b) Arquitectura Long short-term memory bidireccional

Figura 4.4: Operaciones y conexiones de una neurona que forma parte de una arquitectura BLSTM

sino también de la salida producida en el instante anterior, otorgándole una especie de "memoria" de los eventos pasados. Son especialmente interesantes en secuencias donde los atributos son dependientes temporalmente, de forma que la salida en un momento T no depende únicamente de la entrada en T sino de las entradas en T-1, T-2...

Las redes recurrentes son por tanto capaces de aprender dependencias temporales para luego explotarlas posteriormente. Sin embargo, tienen serios problemas a la hora de aprender dependencias a largo plazo. Es aquí donde surgen las redes LSTM, que son redes recurrentes con una serie de modificaciones que les permiten aprender este tipo de dependencias. Su punto clave es el empleo de una célula y tres "puertas": la célula representa el estado interno de la neurona, podría considerarse como la memoria de ésta, y las tres puertas determinan el flujo de datos dentro de la neurona. La puerta de entrada controla la información que entra al estado interno (la célula), en contraposición a esta, la puerta de salida es la encargada de gestionar la información que sale de este estado interno y finalmente la puerta de olvido (*forget gate*) es la que controla la información que se mantiene en el estado interno entre iteraciones (decide qué recordar). Manipulando estas puertas, se es capaz de recordar lo que se necesita y olvidar lo innecesario, permitiendo almacenar dependencias a largo plazo. En la figura 4.4(a) se puede encontrar una representación más detallada de una neurona LSTM.

Aunque en este trabajo no se entrará en el funcionamiento en detalle de una arquitectura LSTM más allá de esta breve explicación, sí que se centrará en la importancia de esta capacidad de aprendizaje del contexto temporal memorizando las dependencias pasadas. No obstante, en las secuencias que aquí se emplean las dependencias no son únicamente hacia el pasado, ya que un vector de coeficientes no depende sólo de los vectores anteriores sino también de los posteriores (la parte intermedia de una palabra depende de la misma manera de la parte final y de la parte inicial). Es decir, hay una dependencia temporal bidireccional, tanto hacia al pasado como al futuro. Por tanto, se emplea una arquitectura LSTM bidireccional (BLSTM) en la que la mitad de las neuronas reciben la secuencia normal y la otra mitad reciben la secuencia inversa y posteriormente se concatenan, así se memorizan las dependencias en ambos sentidos. Un diagrama de la arquitectura BLSTM se puede encontrar en 4.4(b).

Para la realización de los experimentos se han empleado dos capas ocultas BLSTM y una tercera capa *feedforward* que constituye la salida hacia la capa intermedia y que supone el nexo entre las dos partes de la red. Todas las capas emplean la función de activación sigmoidal (se probó la función ReLU pero no dió buenos resultados) y están compuestas de un número de neuronas que se ha ido variando en los diferentes experimentos: 128x2 o 256x2 en el caso de las

Capa	Contexto de la capa	Contexto total
Capa 1	[t-2, t+2]	5
Capa 2	$\{t-2, t, t+2\}$	9
Capa 3	$\{t-3, t, t+3\}$	15
Capa 4	$\{t\}$	15
Capa 5	$\{t\}$	15

Tabla 4.2: Arquitectura TDNN y contexto temporal seleccionado por cada capa

BLSTM y 500, 750, 1000 o 1500 en el caso de la capa *feedforward*.

Time delay neural network

La arquitectura *Time delay neural network (TDNN)*, al igual que arquitecturas recurrentes como la LSTM, puede modelar las dependencias temporales a largo plazo como las que suceden entre eventos acústicos, por lo que cobra especial interés para aprender dichas dependencias a partir de características a corto plazo como los MFCCs [30]. Por otro lado, mientras que las redes recurrentes al tener retroalimentación en sus conexiones no pueden aprovechar la paralelización en el cálculo de la misma manera que las redes *feedforward*, las TDNN evitan esta limitación al tratarse de redes de este tipo, por lo que son más ágiles en los cálculos que las primeras siempre que sea posible la paralelización.

A diferencia de las DNN "estándar" en las que al procesar un contexto temporal amplio la capa inicial aprende una transformada afín para todo el contexto, en las arquitecturas TDNN las transformadas iniciales se realizan sobre contextos más acotados y según se va profundizando en las capas éste se va ampliando, por lo que las capas más profundas tienen la capacidad de aprender relaciones temporales a más largo plazo. En definitiva, cada capa tiene la capacidad de operar a una resolución temporal incremental y diferente.

Mientras que una unidad "estándar" calcula su activación pasando la suma ponderada de los valores de los N atributos (o de las activaciones de la capa anterior) por su función de activación, en el caso de una unidad con retraso temporal se incluyen también los valores de los T instantes anteriores, por lo que la unidad recibe (T+1) x N entradas. Para el ámbito de este trabajo, donde las dependencias temporales son en los dos sentidos (y por ello aplicamos una LSTM bidireccional en lugar de una simple en el apartado 4.2.1) es interesante no sólo incluir instantes anteriores sino también posteriores, de esta forma la red es capaz de aprender relaciones hacia el pasado pero también hacia el futuro. La cantidad de instantes incluidos en el contexto a tener en cuenta es específico de cada capa, por lo que es necesario especificarlo junto a otros parámetros como el número de neuronas de la capa o la función de activación empleada en el momento de definición de la red.

Para este trabajo se ha seleccionado una arquitectura que ya ha demostrado ofrecer un muy buen resultado en la tarea del reconocimiento de locutor [3]. Se emplearán tres capas ocultas de tipo TDNN y dos capas ocultas de tipo *feedforward* antes de la capa de agrupación de estadísticos. Cada una de las tres capas TDNN tomará un contexto temporal de diferente dimensión (véase tabla 4.2) de forma que al final se tome un contexto global de 15 instantes temporales, en el que al emplear MFCCs tomados cada 10 milisegundos estará tratando representaciones sobre locuciones de 150 milisegundos. Las últimas dos capas no toman contextos adicionales, sino que simplemente realizan una transformación de los resultados de la parte con retraso temporal. Estas capas estarán formadas por un número variable de neuronas en función del experimento realizado (entre 128 y 1500) y emplearán como función de activación la sigmoide binaria (véase figura 2.2(a)).

4.2.2. Parte a nivel de locución

La salida de la parte a nivel de ventana es un conjunto de tantos vectores como ventanas haya en una secuencia (en este caso el valor típico es 300 ventanas), teniendo cada uno de estos vectores tantos valores como dimensión tenga la última capa de esta parte de la red (en este caso entre 50 y 1500). La capa intermedia realiza un cálculo de la media y la desviación típica de cada uno de los valores a lo largo del tiempo y al final obtiene un único vector el doble de grande que cada uno de los vectores anteriores. Este vector es la entrada de la segunda parte de la red, y como se puede apreciar, ya no tiene relación con vectores anteriores o posteriores, ya que cada uno de éstos es la representación de una locución completa por sí misma.

Por tanto, en esta segunda parte de la red no se emplean tipos de arquitecturas que exploten las dependencias temporales con su contexto sino redes de tipo *feedforward* en su lugar, para aplicar transformaciones útiles a la tarea de identificación de locutor sobre este nuevo vector de entrada. Como se verá en el apartado 4.3, será de esta parte de la red de donde se extraigan los embeddings, debido precisamente a que trabaja ya sobre la locución completa.

Por su buen resultado en trabajos equivalentes, dentro de esta parte se han empleado dos capas ocultas de 50, 256 o 512 neuronas, y se han realizado experimentos empleando como función de activación la sigmoidal binaria o una función lineal (equivalente a no aplicar activación ya que $f(x) = x$). Como capa de salida contamos con una capa con tantas neuronas como locutores distintos tenemos en nuestra base de datos de entrenamiento, que son 1124 en el caso de limitar a 13 locuciones y 2539 en el caso de limitar a 5 locuciones. Esta capa de salida tendrá como función de activación la función *"softmax"*, que es una generalización de la función logística empleada para comprimir un vector K-dimensional de valores reales en otro vector K-dimensional de valores reales en el rango [0, 1].

Además, si sumamos todos los valores del nuevo vector obtenemos la unidad. La función *"softmax"* puede ser utilizada para representar una distribución de probabilidad sobre K diferentes salidas, y en nuestro caso se interpreta como la probabilidad de que dicho audio pertenezca a cada uno de los locutores distintos (probabilidad a posteriori). La función *"softmax"* como última activación se suele emplear (y en nuestro caso se emplea) conjuntamente con una función de coste[4] de entropía cruzada categórica, por lo que todos los objetivos deben ser convertidos a categóricos antes de ser usados en el entrenamiento.

4.2.3. Otras técnicas aplicadas en entrenamiento

Además de las diferentes modificaciones específicas de cada una de las dos partes de la red enfocadas a agilizar, facilitar o mejorar el entrenamiento y aumentar el rendimiento de la red, hay una serie de medidas que son generales a todo el modelo que describimos en esta sección y que afectan tanto a la parte a nivel de ventana como a la parte a nivel de locución.

Truncado de la longitud de las locuciones

Como se indicaba en el apartado 3.2.3, la longitud de las locuciones es muy variable, y hay que aplicar medidas correctoras en este sentido por dos razones. La primera es que si queremos realizar un entrenamiento ágil y eficaz debemos realizar entrenamiento por lotes (matriz), pasando conjuntos de locuciones que son procesados de manera paralela por la red para luego realizar una única actualización de los pesos tras haber visto todos los ejemplos del lote. Dentro de un

[4]Forma de calcular el error, en sistemas de aprendizaje supervisado, indica la forma de calcular cuánto se diferencian el objetivo con la salida de la red. Ejemplos de funciones de coste pueden ser el error cuadrático medio o el total, o la entropía cruzada

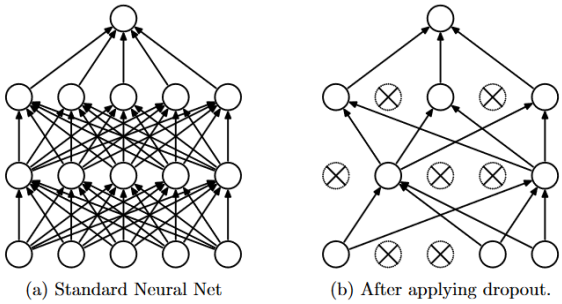


Figura 4.5: Ejemplo de aplicación de Dropout en una red neuronal profunda. Figura extraída de [4].

mismo lote todos los ejemplos deben tener la misma longitud, por lo que deben ser truncados a un tamaño fijo. En nuestro caso empleamos un lote (*batch*) de tamaño 200, aunque en algunos experimentos por razones de restricción de memoria se ha tenido que reducir a 100.

Para truncar las locuciones, lo que se hará es seleccionar un número determinado de vectores de MFCCs consecutivos de manera aleatoria dentro de toda la secuencia de vectores asociada a cada locución, y aquellos que no llegan a esta cantidad, serán completados con ceros. Esta técnica conlleva una ventaja, y es que al seleccionar un fragmento aleatorio de un determinado número de vectores cada época, los datos que recibe la red tienen una alta probabilidad de ser diferentes cada vez, de forma que aumentamos la cantidad de datos "distintos" que se reciben durante el entrenamiento (y así evitamos problemas como el sobre-entrenamiento).

El número de vectores seleccionados es un parámetro que debe ser ajustado y para el que se realiza un análisis en el apartado 5.

Dropout

Un fenómeno que se puede producir en el entrenamiento de redes neuronales y debe ser evitado se conoce como sobre-entrenamiento. Cuando éste se produce, se puede observar una significativa diferencia entre el rendimiento obtenido con el conjunto de datos de entrenamiento y con el conjunto de validación. Esto se debe a que la red está memorizando los patrones que recibe y no tiene la suficiente capacidad de generalización como para clasificar correctamente patrones similares pero no iguales a los vistos.

Una de las técnicas más usadas para evitar el sobre-entrenamiento se conoce como *"Dropout"*[4] y consiste en realizar la acción que su propio nombre indica: abandonar o retirar algunas neuronas (tanto las visibles como las ocultas) de una red neuronal. Lo que se hace es ignorar unidades (seleccionadas de manera aleatoria) durante la fase de entrenamiento, por lo que dichas unidades no son consideradas en las fases de propagación hacia delante y hacia atrás (no se produce tampoco una actualización de sus pesos). Más técnicamente, lo que se hace es ignorar cada una de las unidades en una época con probabilidad (1-P) o mantenerla con probabilidad P, y este proceso se repite por cada época de entrenamiento. Una vez hemos completado el entrenamiento, para la fase de explotación de la red se emplean todas las neuronas sin ignorar ninguna, pero sus pesos deben ser ajustados para compensar este aumento del número de neuronas. Por tanto, los pesos de todas las conexiones se multiplican por la probabilidad de *"Dropout"* P. Un ejemplo de aplicación de *"Dropout"* sobre una red neuronal profunda se puede observar en la figura 4.5.

El *"Dropout"* hace que cada neurona deba aprender con un conjunto seleccionado aleatoriamente de otras neuronas, lo que la hace más robusta y la conduce a crear características útiles

por su cuenta sin apoyarse en otras neuronas que puedan corregir sus "errores". Así, conseguimos generalizar mejor ya que aunque las co-adaptaciones complejas pueden ser entrenadas para trabajar bien sobre el conjunto de entrenamiento, sobre datos no vistos tienen más probabilidad de fallar que muchas co-adaptaciones simples.

Esta técnica se ha aplicado en este trabajo únicamente a las capas a nivel de ventana (con una P igual a 0.75), y se realizará una comparación entre los resultados sin su empleo o tras aplicarla.

Inclusión de ruido Gaussiano

Otra técnica que trata el problema del sobreajuste o sobre-entrenamiento es la inclusión de ruido. Cuando debido al tamaño reducido de una base de datos en lugar de aprender las relaciones entre entradas y salidas la red memoriza los parámetros de entrada y sus salidas correspondientes (y no es posible aumentar el tamaño de la base de datos), modificar los datos empleados incluyendo ruido generado aleatoriamente permite dificultar esta memorización (ya que los ejemplos de entrenamiento cambian cada época debido a la aleatoriedad) y facilita la generalización.

Esta técnica tiene un efecto de regularización y aumenta la robustez del modelo. De hecho, expande el tamaño de la base de datos al suministrar entradas diferentes en cada época (es una forma simple de *"Data augmentation"*).

En este trabajo se analizan experimentos incluyendo o no ruido gaussiano centrado en cero y con desviación típica de 0,2.

4.3. Extracción de embeddings

En esta tercera fase se ha empleado la API funcional Keras junto con TensorFlow para la extracción de los embeddings asociados a cada una de las locuciones disponibles. Por otro lado, se ha empleado Matlab para operaciones auxiliares sobre matrices de cara a la preparación de los datos para la siguiente fase.

Una vez ya se ha entrenado el extractor el siguiente paso es emplearlo. Como ya se mencionó anteriormente, los embeddings son representaciones de longitud fija de una locución completa y específicas para una tarea determinada. Es por esto que hemos entrenado una red para que sea capaz de extraer vectores de características con información útil para el reconocimiento de locutor (véase apartado 4.2). Por tanto, lo que se haría a continuación es introducir a la red cada uno de los ficheros de los que disponemos en la base de datos, de forma que al final obtenemos el locutor cuya autoría es más probable para dicho fichero. Sin embargo, esta selección de locutor no será empleada ya que no es nuestro objetivo final. En su lugar extraeremos las activaciones de algunas de las capas ocultas que se encuentran en la parte a nivel de locución, y que contienen una transformación de los datos de entrada especialmente enfocada a obtener toda la información útil para poder identificar la autoría de la locución. Es por ello que hay que poner especial cuidado en la selección del número de neuronas de estas capas, ya que marcan la dimensión de los embeddings extraídos.

Como indicamos en el apartado 4.2.2, se han usado dos capas ocultas y en función de si seleccionamos la capa más lejana a la salida o la más cercana obtendremos una representación de la locución con un mayor o menor nivel de abstracción, respectivamente.

En los experimentos realizados en este trabajo de fin de grado se han analizado en profundidad los distintos parámetros que afectan a esta fase. Se ha comparado el rendimiento con

distintos tamaños de embeddings (50, 256, 325 y 512) y con distintas activaciones en las capas ocultas de la segunda parte de la red (función sigmoidal o lineal), y también se ha analizado el rendimiento en función de si los embeddings se extraían de la primera capa, de la segunda o realizando una concatenación de ambas.

En definitiva, el objetivo de este apartado es crear una base de datos similar a la anterior, pero en lugar de tener representaciones de locuciones en formato de matrices de MFCCs, donde la longitud de la locución influye en el tamaño de la matriz, tener representaciones de locuciones en formato embedding. Al tener cada locución en un vector de la misma longitud, estos pueden ser guardados como una única matriz y el modelado y tratamiento de los datos es mucho más sencillo. Además, ahora se cuenta con la ventaja de que se pueden aplicar técnicas de análisis factorial como las que se ven en el siguiente apartado.

4.4. Entrenamiento y evaluación de modelos GPLDA y LDA

Una vez ya se dispone de la base de datos con los embeddings, se está en disposición de realizar la verificación de locutor. Esta tarea consistirá en comparar dos embeddings para decidir si provienen del mismo locutor o no. De uno de ellos se conoce su autor (el modelo) y el otro es el audio a verificar (test). Por tanto, lo que hay que hacer es extraer alguna medida de similitud para que se emplee como criterio para determinar si la autoría del audio es legítima o no.

Además, al haber obtenido una representación de cada audio en forma de vector con una dimensionalidad reducida, se pueden utilizar estrategias convencionales de compensación de variabilidad, como el análisis discriminante lineal o el análisis gaussiano discriminante probabilístico lineal.

En esta sección se detallarán las distintas técnicas de tratamiento de los embeddings, la medida de similitud empleada para la comparación entre éstos y las métricas de error para la evaluación de los distintos sistemas, que nos permiten comparar el rendimiento de los diferentes experimentos.

4.4.1. Análisis discriminante lineall

El análisis discriminante lineal (*"Linear Discriminant Analysis"*, LDA) es una técnica comúnmente empleada en reconocimiento de patrones que busca encontrar combinaciones lineales entre las características de los embeddings para facilitar la discriminación entre múltiples clases y minimizar la variabilidad intra-clase. Para ello, encuentra direcciones ortogonales en el espacio de características que son más efectivas en esta discriminación y proyecta las características iniciales en estas direcciones para mejorar la discriminación.

Para poder encontrar la transformación que maximiza la capacidad de discriminación es necesario entrenar el algoritmo. Para ello se calculan las matrices de covarianza entre-clases y intra-clase (empleando los mismos ficheros que los que se usaron para entrenar el extractor) y el algoritmo de optimización busca maximizar la varianza entre-clases mientras minimiza la varianza intra-clase. El resultado es un vector de una dimensión menor al original (la dimensión seleccionada es un parámetro a analizar) con las combinaciones lineales de las características originales en función de las direcciones ortogonales obtenidas.

4.4.2. Análisis gaussiano discriminante probabilístico lineal

El análisis gaussiano discriminante probabilístico lineal (GPLDA) comparte con el LDA el objetivo de encontrar combinaciones lineales entre las características para facilitar la discrimina-

ción entre clases. Y, al igual que ésta, entrena una proyección de características que le permita maximizar la variabilidad entre-clases mientras minimiza la variabilidad intra-clase.

Consiste en realizar las mismas suposiciones que al aplicar *"Joint Factor Analysis"* (un vector contiene variabilidad dependiente del locutor y variabilidad dependiente del canal o sesión, ambas residiendo en subespacios de menor dimensión). De hecho, las ecuaciones de modelado son idénticas, aunque la principal diferencia entre ambas técnicas es que JFA modela los supervectores de GMM mientras que GPLDA modela representaciones de locuciones completas como los embeddings o los i-vectors, y por tanto las condiciones son diferentes.

4.4.3. Similitud coseno

Para algunas pruebas, como medida de similitud entre dos vectores se ha empleado la similitud coseno, que devuelve el coseno del ángulo formado entre ellos. Se emplea el coseno porque devuelve el valor máximo cuando el ángulo formado entre los vectores es 0 (cuando se entiende que son más similares) y el valor mínimo cuando el ángulo formado es máximo (cuando se diferencian en 180 grados que entonces se entiende que son totalmente diferentes).

Esta medida muestra la medida de similitud como un valor, lo que facilita el criterio de decisión (por ejemplo un umbral), y devuelve un valor acotado tanto por arriba como por abajo y centrado en cero.

Por tanto, para decidir sobre si un locutor de test es legítimo o no, simplemente deberemos fijar un umbral entre -1 y 1 y asignar como legítimas todas aquellas comparaciones que queden por encima del umbral y como no legítimas, el resto.

4.4.4. Equal Error Rate

Para realizar una medición del rendimiento del sistema emplearemos una lista de pruebas que nos indica qué dos ficheros comparar y si pertenecen al mismo locutor o no. En este tipo de tareas es mucho más común que el locutor no sea legítimo (non-target) a que sea legítimo, por lo que si empleamos una métrica de error que no tenga en cuenta esto como puede ser el porcentaje de acierto, con dos sistemas igual de aleatorios (como pueden ser afirmar siempre locutor legítimo o siempre locutor no legítimo) estamos obteniendo métricas muy diferentes y no ajustadas a la realidad. Una métrica muy utilizada para estos casos (es una de las medidas más utilizadas en reconocimiento de locutor) es el *"Equal Error Rate"* (EER). El EER nos dice el porcentaje de error cuando el umbral de decisión hace que las tasas de falsa aceptación y falso rechazo se igualen. Por tanto, esta métrica es más robusta ante desequilibrios en la cantidad de comparaciones rechazadas y aceptadas y no depende del umbral de decisión seleccionado.

El EER será la métrica empleada en nuestro trabajo para seleccionar cuál es el umbral de decisión del valor de la similitud coseno y para comparar los rendimientos entre los distintos experimentos y entre las distintas técnicas de análisis factorial dentro de un mismo experimento.

5

Experimentos y resultados

En el apartado anterior han quedado indicados todos los parámetros que deben ser ajustados para tratar de mejorar el rendimiento del sistema. En este apartado se expondrán los resultados más relevantes de los experimentos realizados con una configuración u otra de estos parámetros y se realizará un estudio sobre las configuraciones que muestran mejores resultados. La medida empleada para el estudio será el *"Equal Error Rate"*, expuesta en el apartado 4.4.4.

5.1. Análisis de la configuración de la parte a nivel de trama de la red

En este apartado realizaremos un estudio de las distintas arquitecturas posibles para la parte a nivel de ventana del extractor. Compararemos las dos arquitecturas principales (TDNN contra BLSTM) así como distintos tamaños de capa que hemos empleado.

Para el caso de la arquitectura BLSTM se han empleado tres configuraciones principales variando el número de neuronas de las dos capas ocultas LSTM y de la tercera capa *feedforward*. Por otra parte, para la arquitectura TDNN se han empleado dos versiones diferentes, una con 128 neuronas en las cuatro primeras capas (las tres primeras con retraso temporal) y otra con 256. La última capa en ambos casos tendrá 1500 neuronas.

El resto de parámetros se han dejado fijos e iguales entre pruebas, de forma que los experimentos puedan ser comparados en igualdad de condiciones. Se han empleado embeddings de 512 valores, sin aplicar dropout, realizando filtrado de locutores[1], con una longitud de los ficheros de audio de entrenamiento truncada a tres segundos (véase apartado 4.2.3) y la función de activación sigmoidal. GPLDA y LDA han sido aplicados con 400 dimensiones.

En la tabla 5.1 se puede ver una comparación de los resultados sin haber aplicado ninguna técnica de transformación del vector (Cos. score), y aplicando LDA y GPLDA. Lo primero que se puede observar es que se obtienen errores en el entrenamiento de GPLDA con grandes dimensionalidades (siempre ocurre cuando la última capa es 1500), lo que nos da la intuición de que los valores son demasiado dependientes entre sí, haciendo que el cálculo de autovalores y

[1]El filtrado de locutores consiste en seleccionar sólo 13 ficheros de aquellos locutores que tengan un número igual o mayor de locuciones. Aquellos que tienen menos son descartados.

Sistema	**EER**		
	Cos. score	**LDA**	**GPLDA**
LSTM (256, 1500)	41,9	27,8	*
BLSTM (128x2, 500)	**25,7**	**22,6**	23,2
BLSTM (175x2, 1000)	30,23	26	32,95
BLSTM (256x2, 1500)	39,4	33,8	68,08*
TDNN (128, 1000)	31,5	25,85	22,8
TDNN (256, 1500)	30,9	26,3	**22,5**

El primer valor dentro del paréntesis indica el número de neuronas en las capas BLSTM o TDNN y el segundo el número de neuronas en la última capa *"feedforward"*. El asterisco indica que en el entrenamiento de GPLDA han habido errores debido a que la matriz de transformación obtenida tiene un determinante muy cercano a cero (no invertible) o está mal condicionada, lo que provoca que se obtengan autovalores negativos y la transformada obtenida no sea correcta.

Tabla 5.1: Comparativa de resultados para variaciones de arquitecturas del nivel de trama de la red

autovectores no se pueda realizar (ya que se necesita invertir la matriz, y esto no se puede hacer cuando su determinante es cero).

Si analizamos los resultados, podemos ver que para las tres técnicas en el caso de BLSTM se obtiene un mejor resultado con arquitecturas pequeñas, lo que unido al error anterior nos da la intuición de que para nuestro problema es mejor comprimir la información a nivel de locución en vectores pequeños. Por otro lado, se puede ver que esta influencia del tamaño de arquitectura no es tan notable para la estructura TDNN, que obtiene un resultado similar independientemente del número de neuronas empleado. Es interesante remarcar que aunque la BLSTM en su configuración más pequeña (128x2, 500) consigue el mejor rendimiento para similitud coseno y LDA (25,7 % y 22,6 % de EER respectivamente), es la TDNN la que obtiene un mejor rendimiento tras aplicar GPLDA y un mejor rendimiento general (22,5 %), de lo que se puede deducir que efectivamente las técnicas que explotan el contexto general son adecuadas para este tipo de tramas. Finalmente, cabe destacar el pobre rendimiento de la LSTM no bidireccional, lo que nos confirma que la dependencia temporal de la señal de audio de las locuciones se encuentra tanto a futuro como a pasado.

5.2. Análisis de la inclusión de Dropout

Si analizamos el porcentaje de acierto en el conjunto de entrenamiento y validación en términos del error en la clasificación, podemos ver que por lo general son muy diferentes, habiendo una diferencia del 23 % de media entre todos los experimentos. Por tanto, es posible que nos encontremos con un caso de sobre-entrenamiento ya que el resultado con los patrones vistos es claramente mejor que con los no vistos. Como se vio en la sección 4.2.3, el *"Dropout"* es una medida efectiva para contrarrestar el sobreentrenamiento, por lo que en este apartado analizaremos la mejora que supone esta técnica en el resultado final. En la tabla 5.2 se puede ver una comparación con uno de los modelos que mejor resultado ha obtenido en el experimento anterior sin aplicar esta técnica y tras aplicarla. En este caso se ha aplicado *"Dropout"* a todas las capas de la primera parte de la red.

Como se puede ver en la tabla, los resultados son prácticamente iguales y el dropout no está siendo útil en este caso. Esto nos sugiere que el sobre-entrenamiento no es debido a que se crean dependencias entre neuronas para la extracción de características, sino que podría ser por la

Sistema	**EER**		
	Cos. score	**LDA**	**GPLDA**
BLSTM (128x2, 500) Sin dropoput	25,7	**22,6**	23,2
BLSTM (128x2, 500) Con dropout	**25,5**	22,9	23,2

Tabla 5.2: Análisis de la aplicación de dropout

Sistema	**EER**		
	Cos. score	**LDA**	**GPLDA**
BLSTM (128x2, 500) Datos balanceados	25,7	25,4	17,9
BLSTM (128x2, 500) Datos no balanceados	**24,46**	**24,57**	**14,6**
TDNN (256, 1500) Datos balanceados	30,9	26,2	19,8
TDNN (256, 1500) Datos no balanceados	**29,63**	**25,8**	**14,81**

Tabla 5.3: Análisis del efecto del balanceo de los datos en el resultado del extractor de características

cantidad de datos empleados en este trabajo en comparación con otros trabajos enfocados a la misma tarea. Por ello, cobra sentido pensar que es necesaria una ampliación de la base de datos empleada.

5.3. Análisis de la limitación del número de ficheros por locutor

Uno de los primeros tratamientos que se hizo a los datos fue limitar la cantidad de audios por locutor a 13, quitando todos los que pasen de este número y eliminando a todos los locutores que no lleguen a esta cantidad. Se hizo esto para balancear los datos por clase ya que unos datos muy desbalanceados pueden llevar a errores en la clasificación por tener "preferencias" por una clase. Sin embargo, nuestra tarea final es la de verificar un locutor, y la clasificación de la red es una tarea auxiliar que simplemente nos permite entrenar la red para que sea capaz de extraer una buena representación de las características discriminativas del autor. Por tanto, la presencia de datos desbalanceados no afecta de forma tan directa al resultado final, ya que igualmente se sigue entrenando la red para extraer este tipo de características (aunque aprende a extraer de un fragmento de audio las características relacionadas con su autor de forma desigual, aprendiéndose más unos autores que otros, sigue aprendiendo los mecanismos generales de extraer a partir de un fragmento de audio las características de su autor, que son comunes a todos ellos). Además, ya en el apartado anterior se mencionó la posibilidad de aumentar la cantidad de datos disponibles para mejorar el rendimiento del sistema, y a falta de más datos y antes de aplicar técnicas de *"Data augmentation"* más complejas, hemos experimentado utilizando más datos dentro de los disponibles.

En la tabla 5.3 se realiza una comparación con los dos sistemas que mejores resultados han dado hasta el momento, TDNN (256, 1500) y BLSTM (128x2, 500). Los parámetros aplicados son los siguientes: embeddings de 512 valores, sin dropout, dimensión de LDA y GPLDA de 30 dimensiones, función de activación sigmoidal y longitud de los audios truncada a 3 segundos. Por un lado, se han empleado estrictamente 13 ficheros por locutor (datos balanceados, 14612 ficheros, 1124 locutores), por el otro, se han incluido todos los locutores con cinco o más ficheros y sin limitar el número de ficheros de los que disponen (datos no balanceados, 34093 ficheros, 2539 locutores).

Como se puede observar en la tabla, emplear un mayor número de datos aunque estén desbalanceados aumenta el rendimiento en todos los casos y para ambos modelos, y aunque para la similitud coseno o el LDA el aumento de rendimiento es leve, para GPLDA es muy

Número de neuronas por capa	**EER**		
	Cos. score	**LDA**	**GPLDA**
50	28,7	26,4	21,9
256	28,1	26,1	19,8
512	**25,6**	**24,2**	**18,2**

Tabla 5.4: Comparativa de variaciones de arquitecturas de la segunda parte de la red

significativo, disminuyendo un 3,3 % y un 5 % el porcentaje de error para la BLSTM y la TDNN, respectivamente. Esta mejora general de rendimiento permite afirmar dos cosas: la primera es que efectivamente se confirma la intuición de que era necesario un aumento de los datos disponibles (dejando la puerta abierta a aplicar en un futuro técnicas de *"Data augmentation"*). Y la segunda es que todo el perjuicio al rendimiento que puede introducir el desbalanceo de los datos se ve claramente compensado con las ventajas de aumentar la cantidad de ficheros y de locutores disponibles.

5.4. Análisis de la configuración de la parte a nivel de segmento de la red

En este apartado se realizará un estudio de distintas topologías posibles para la parte a nivel de locución del extractor de embeddings. En este caso no se variará el tipo de red empleada sino que se analizará la dimensión del vector embedding que mejor rendimiento consigue. Por otra parte, comprobaremos cuál de las dos capas es la idónea para la extracción, o si por el contrario lo adecuado es realizar una concatenación de ambas. En la tabla 5.4 podemos ver una comparativa entre distintas dimensiones de las capas a nivel de locución para la arquitectura que mejor rendimiento ha conseguido en el apartado anterior, la arquitectura BLSTM (128x2, 500). Los parámetros de los experimentos son los mismos que en el apartado de análisis de la arquitectura de la primera parte de la red, con la diferencia de que ahora LDA y GPLDA tendrán 40 dimensiones en lugar de 400.

Como se puede ver en esta tabla, el mejor rendimiento en los tres casos se obtiene cuando empleamos 512 neuronas en las capas de embeddings. Este resultado es el esperado ya que en trabajos similares [3][22] esta dimensionalidad también resulta ser la óptima. Además, coincide con la dimensión de los i-vectors, que tienen entre 400 y 600 valores. Cabe destacar la diferencia con los resultados obtenidos en el apartado de análisis de la primera parte de la red, donde se nos muestra que (al menos en el caso de las BLSTM) una arquitectura más pequeña es lo óptimo.

Por otro lado, también es interesante analizar qué capa es la óptima para extraer los embeddings. Si extraemos los embeddings de la capa más lejana a la salida estaremos obteniendo una representación más lejana a la tarea de clasificación que si la tomamos de la capa más cercana. Otra opción es concatenar ambas, con lo que tendríamos una representación con dos niveles de abstracción diferentes. En la tabla 5.5 se comparan los rendimientos según obtenemos los embeddings de la capa más lejana a la salida (A), de la capa más cercana (B) o concatenando ambas. No se han modificado los parámetros empleados con respecto al experimento anterior (con la diferencia de que ahora no se balancean los datos) y como modelo se ha seleccionado la red TDNN (256, 1500) por su buen rendimiento empleando esta muestra de datos no balanceados.

Como se puede observar, los resultados obtenidos para las tres técnicas son mejores en el caso de tomar la salida de la capa A. Esto nos indica que una representación más alejada de la clasificación, aunque ya transformada para la tarea de identificación de locutor, es más eficiente que una representación más centrada en la clasificación posterior. Por otro lado, los resultados nos sugieren dos posibilidades, que distintos niveles de abstracción no son complementarios sino

Capa seleccionada	**EER**		
	Cos. score	**LDA**	**GPLDA**
Capa A (más lejana a salida)	**29,63**	**25,8**	**14,81**
Capa B (más cercana a salida)	30,4	25,97	16,43
Concatenación de ambas	30,08	26,76	16,81

Tabla 5.5: Comparacion de rendimiento en funcion de la capa de extraccion

Longitud de los segmentos de audio	**EER**		
	Cos. score	**LDA**	**GPLDA**
3 segundos	29,63	25,8	**14,81**
20 segundos	**29,46**	**22,88**	15,47

Tabla 5.6: Comparativa de resultados en función de la duración de los fragmentos de audio

más bien contraproducentes, ya que en conjunto obtienen peores resultados que por separado, o que la dimensión de la unión de ambos es demasiado grande.

5.5. Análisis de la limitación de la longitud de las locuciones en entrenamiento

Además del desajuste de la cantidad de locuciones entre los distintos autores, nuestro dataset contiene ficheros de una longitud muy dispar. Precisamente con este sistema lo que se busca es obtener un método de representación con una dimensión independiente de la duración de los fragmentos de audio. Sin embargo, como ya vimos en el apartado 4.2.3, para entrenar este extractor de manera ágil es necesario truncar todos los ficheros a una dimensión determinada. En este apartado se realiza una comparativa entre el resultado tras el truncado a una duración pequeña (3 segundos) o a una duración grande (20 segundos). Hay que destacar que en este caso no se eliminan ficheros, sino que se completan con ceros en caso de no llegar a la longitud fijada. Se podría pensar que esta alteración de los datos podría tener un efecto negativo, pero con el conjunto de datos que empleamos en esta parte de los experimentos (conjunto de datos no balanceado) los ficheros de menos de tres segundos son 24 (0,083 % del total) y de menos de 20 segundos son 69 (0,24 %) por lo que su alteración no es apenas significativa con respecto al ámbito global.

Las pruebas se han realizado sobre el modelo TDNN (256, 1500), con 512 valores en los embeddings extraídos de la capa A, 30 dimensiones para LDA y GPLDA, activación sigmoidal binaria, sin dropout y con la base de datos no balanceada.

Como se puede ver en la tabla 5.6, la diferencia, aunque algo mayor en el caso de LDA, no es muy grande entre ambos escenarios. Esto nos indica que no es un factor muy significativo dentro del rendimiento general del sistema. Sin embargo, se produce un hecho interesante y es que aumentar la longitud del fichero resulta beneficioso en el caso de no aplicar ninguna técnica de análisis factorial y especialmente beneficioso si se aplica LDA, mientras que para el GPLDA es perjudicial. Sería necesario un análisis con mayor profundidad en este aspecto para poder comprender cómo las diferencias de funcionamiento de ambas técnicas explotan de una forma más o menos óptima un marco temporal mayor o menor.

En nuestro caso, como hasta ahora la técnica GPLDA ha mostrado un mayor rendimiento que LDA o el cálculo directo de la distancia coseno en todos los modelos probados y para todas las configuraciones de parámetros, consideramos más interesante el empleo de fragmentos de audio de tres segundos.

Dimensión del vector de salida de LDA y GPLDA	**EER**		
	Cos. score	**LDA**	**GPLDA**
20 valores	25,7	26,5	18,4
30 valores		25,4	17,9
40 valores		24,2	18,2
150 valores		22,1	22,5
400 valores		22,6	23,2

Tabla 5.7: Rendimiento del sistema en función de la dimensión de los sistemas LDA y GPLDA

5.6. Análisis de la dimensionalidad de los métodos LDA y GPLDA

Por los experimentos realizados en el apartado 5.4, se puede afirmar que una dimensión grande de los embeddings (de por ejemplo 512 valores) obtiene mejores resultados que una dimensión pequeña (de por ejemplo 50 valores). No obstante, para las técnicas LDA y GPLDA la salida del extractor de embeddings no es la representación final empleada para la verificación de locutor, sino que pasa por una transformación previa en la que se calculan combinaciones lineales de los valores (véase apartado 4.4). Por tanto, la dimensión del LDA y GPDLA marca cuántos valores y cuáles van a ser empleados para obtener las métricas de similitud entre locutores y por tanto es un parámetro que debe ser ajustado. En la tabla 5.7 se muestra una comparativa para diferentes dimensiones con el modelo BLSTM (128x2, 500), sin dropout, datos balanceados, embeddings de 512 valores extraídos de la capa A, activación sigmoidal binaria y truncado de los audios a tres segundos.

De los resultados de esta tabla podemos deducir que la dimensión efectivamente juega un papel decisivo en el rendimiento general del sistema, con diferencias de hasta casi un 5 %. Sin embargo, este aprovechamiento es diferente en función de si se emplea LDA o GPDLA. Si nos centramos en los resultados obtenidos con LDA podemos ver que el resultado va mejorando a medida que aumentamos la dimensión, hasta llegar a su máximo en 150. Por otra parte, con GPLDA el rendimiento mejora en la otra dirección, aumentando a medida que disminuimos esta dimensión y alcanzando su máximo en 30.

5.7. Comparación con i-vectors

Una vez ya hemos realizado el análisis de cuáles son los parámetros óptimos, podemos comparar el rendimiento de este modelo con las técnicas que forman parte del estado del arte. En concreto, la técnica más usada en los últimos años y que constituye un buen sistema de referencia con el que comparar los embeddings por su buen rendimiento en el reconocimiento de locutor y de idioma es el modelo de variabilidad total (véase apartado 2.1.2) que da como resultado los i-vectors. La comparación entre los resultados de embeddings e i-vectors es muy simple ya que ambos son representaciones de longitud fija independiente de la longitud del audio, por lo que se calculará de igual manera la distancia coseno y se aplicará LDA y GPLDA para así obtener la métrica EER. La extracción de los i-vectors para todos los ficheros de audio queda fuera de este trabajo de fin de grado y ha sido aportada por la tutora. En la tabla 5.8 se puede ver una comparativa del rendimiento entre el mejor resultado obtenido con el extractor de embeddings y el conjunto de i-vectors.

Como se puede apreciar en la tabla, los modelos de variabilidad total superan en todos los casos al extractor de embeddings. Sin embargo, otras publicaciones (como [3] o [22]) muestran los resultados opuestos (siendo mejor el sistema basado en el extractor de embeddings), a pesar de emplear las mismas arquitecturas y parámetros que algunos experimentos de este trabajo.

Modelo	**EER**		
	Cos. score	**LDA**	**GPLDA**
Embeddings	24,46	24,57	14,6
Variabilidad total (i-vectors)	**7,02**	**6,45**	**2,89**

Tabla 5.8: Comparativa del rendimiento entre un sistema basado en embeddings y modelos de variabilidad total

Con todo esto, y unido a las diferencias de rendimiento entre el conjunto de entrenamiento y el de validación, a la imposibilidad de solventar este problema con técnicas contra el sobre-entrenamiento como el dropout o la inclusión de ruido gaussiano, y a la mejora de los resultados al ampliar la base de datos al desbalanzearla, se puede llegar a la conclusión de que es posible que se tenga peor rendimiento no porque la técnica sea peor, sino porque la base de datos empleada no es suficiente o no es la adecuada. Serían necesarias más pruebas con conjuntos de datos similares a los empleados en los experimentos [3] o [22] para poder esclarecer las causas de estas diferencias.

5.8. Modelo con mejor rendimiento

Tras el análisis de todos los parámetros que influyen en el rendimiento del modelo, a continuación se detalla la configuración que ha obtenido mejores resultados:

- **Tipo de red:** Red *"Bidirectional Long Short-Term Memory"* (BLSTM)
- **Arquitectura de la primera parte:** Dos capas de 128x2 neuronas más una tercera capa *"feedforward"* de 500 neuronas. Activación sigmoidal binaria. No se aplica dropout.
- **Arquitectura de la segunda parte:** Dos capas *"feedforward"* de 512 neuronas más la capa de salida (de 2539 unidades). Se utilizan los embeddings de las activaciones de la capa más lejana a la salida.
- **Limitación del número de ficheros por locutor:** 5 ficheros mínimos por locutor, no hay límite superior.
- **Limitación de la longitud de las locuciones:** Audios truncados a 3 segundos.
- **Dimensión LDA y GPLDA:** 30 valores

Con esta configuración se obtiene un resultado del 14,6 % de EER para la tarea de reconocimiento de locutor.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se realiza un sistema de verificación de locutor independiente de texto empleando como base los trabajos del CLSP y el HLTCOE de la John Hopkins University[3] y los trabajos conjuntos de AUDIAS (EPS-UAM, Madrid) con BUT Speech (FIT, Brno)[22]. Al igual que éstos, se busca mejorar el rendimiento obtenido por sistemas basados en variabilidad total, que hasta ahora formaban parte del estado del arte.

A pesar de que se emplean técnicas y arquitecturas similares, no se ha llegado en este TFG al mismo rendimiento de dichos trabajos. Sin embargo, donde sí hay una gran diferencia es en los datos empleados para el entrenamiento y en la cantidad de los mismos, por lo que se ha llegado a la conclusión de que la diferencia de rendimiento se debe a los datos empleados para el entrenamiento del extractor de características y no tanto a las técnicas o modelos empleados.

Más que en el rendimiento obtenido, este trabajo resulta interesante al aportar un análisis de la utilidad de las diferentes técnicas que pueden ser aplicadas a la extracción de embeddings y su impacto en el rendimiento del sistema completo, partiendo de los trabajos en el estado del arte mencionados y modificando las topologías empleadas.

Debido al bajo rendimiento causado por los datos empleados respecto a los i-vectors (y al hecho de que se obtiene una gran diferencia en los resultados de entrenamiento y en los de test), se tiene la hipótesis de que se puede estar produciendo algún tipo de sobre-entrenamiento por lo que se aplicarán diversas técnicas para contrarrestrarlo, como son el Dropout y la inclusión de ruido Gaussiano. En los resultados se pudo ver que no había una diferencia significativa entre el uso o no de estas técnicas por lo que se deduce que aunque es posible que haya sobre-entrenamiento, estas técnicas no son capaces de paliarlo.

Por otra parte, una aportación importante de este trabajo es la comparativa entre tres arquitecturas de red neuronal para la extracción de embeddings como son la LSTM, la BLSTM y la TDNN. Tras realizar distintas pruebas se pudo ver que la BLSTM y la TDNN dan resultados muy similares y claramente superiores a la LSTM (diferencia del 5,2 % de EER en LDA y del 16,2 % en *"Cosine score"* con respecto a la BLSTM para las mismas condiciones), siendo mejor cada una de ellas en diferentes aspectos. Por tanto, se confirma la hipótesis de que las secuencias de MFCCs tienen dependencias temporales en ambas direcciones y al tenerlas en cuenta

se mejoran los resultados. Además, se puede observar que un sistema basado en recurrencias (BLSTM) es igual de válido para esta tarea que un sistema *"feedforward"* (TDNN).

Si nos centramos en la arquitectura de la red, se puede observar que los mejores resultados se obtienen con una dimensión pequeña en las capas de la parte a nivel de ventana (128x2 para BLSTM y 256 para TDNN, aunque en el caso de las TDNN no es tan significativo), y con una dimensión grande en las capas a nivel de locución. De aquí podemos inferir que es necesario comprimir la información cuando se trata de una serie de vectores de MFCCs pero luego cuando ya estamos representando locuciones completas es conveniente transformarla a un espacio abstracto de características de mayor dimensión (512 valores), que además coincide aproximadamente con la dimensión que suelen tener los i-vectors (entre 400 y 600 valores).

En cuanto a los datos empleados, aunque a priori se podría pensar que un balanceo de los datos (mismo número de ficheros por cada locutor) sería beneficioso, los resultados nos muestran que aumentar los ficheros disponibles al emplear todas las locuciones de cada autor (aunque la cantidad de audios por autor sea muy irregular) mejora el rendimiento del sistema en todos los casos (entre un 0,4 % y un 4,99 % en función de la arquitectura empleada). Esta mejora tiene sentido, ya que en la red buscamos que sea capaz de extraer aquellas características que mejor definen a su autor más allá de la función final de clasificación, y en este caso aunque es posible que la red tenga un leve sesgo hacia los autores de los que más audios se disponen, lo que hace a un audio pertenecer a un autor determinado es similar en todos ellos, por lo que aumentar la cantidad de datos a cambio de desbalancearlos es beneficioso (probablemente por la "necesidad" de la red de recibir más datos). No obstante, un desequilibrio muy grande podría hacer que la red acabara tomando como características propias del autor particularidades de los autores más frecuentes, por lo que el desbalanceo es un factor a controlar de cara a futuros experimentos.

Finalmente, se realizó un estudio basado en los resultados obtenidos por las técnicas de LDA y GPLDA en función de la dimensión de los vectores de características que dan como resultado. En la gran mayoría de los experimentos GPLDA obtiene un mejor resultado que LDA, y éste mejora a medida que se reduce la dimensionalidad de los vectores que genera hasta obtener un mínimo del error en torno a una dimensión de 30 valores. No es así para LDA, que reduce su error a medida que la dimensión de sus vectores aumenta, hasta obtener un mínimo en torno a una dimensión de 150. No obstante, el mejor resultado con LDA sigue quedando lejos del mejor resultado con GPLDA, por lo que se puede concluir que en general se preferirán vectores de pequeña dimensión generados a partir de la técnica GPLDA.

6.2. Trabajo futuro

Como ya se ha indicado en el apartado anterior, se llega a la conclusión de que sería necesario un aumento de la base de datos por ser la disponible insuficiente para que este tipo de técnicas sean capaces de llegar al rendimiento de los i-vectors. En este sentido sería interesante incluir otras bases de datos ya extendidas y estandarizadas (como el *Speaker Recognition Evaluation* en otros años u otros conjuntos de datos suministrados por el *Linguistic Data Consortium (LDC)*). Además, también sería útil realizar técnicas de *"Data Augmentation"* que permiten aumentar la cantidad de datos a nuestra disposición sin incluir nuevos fragmentos de audio al realizar distintas transformaciones sobre los audios originales (como la inclusión de ruido, de reverberación, etc.).

Por otra parte, se ha podido observar que las topologías BLSTM y TDNN, aunque basadas en estrategias diferentes, consiguen un resultado similar y complementario (ya que BLSTM da mejores resultados sin aplicar transformaciones a la salida y con LDA mientras que TDNN se aventaja al aplicar GPLDA). Estos resultados, unidos al hecho de que extraen información de contexto temporal de manera diferente, llevan a pensar que un modelo que integre una combinación de ambas podría obtener un mayor rendimiento que con ambos modelos por separado.

También resultaría interesante combinar las estrategias de embeddings e i-vectors para obtener una aproximación conjunta. Por sus diferencias en las formas de generar los vectores de representación, es posible que se extraigan distintos tipos de información sobre el audio original, por lo que una estrategia mixta podría mejorar los resultados con respecto a cualquiera de las dos técnicas por sí solas.

6.3. Puntos de partida para este trabajo

Para la realización de los experimentos detallados en este trabajo de fin de grado se ha contado con la ayuda de las siguientes aportaciones realizadas por la tutora de este trabajo, Alicia Lozano Díez:

- **Extracción de las matrices de MFCCs:** *Scripts* que emplean funcionalidades de Kaldi para extraer las matrices de coeficientes cepstrales en las frecuencias de Mel a partir de los ficheros de audio. En este sentido, también ha proporcionado la funcionalidad de filtrado de locutores en función de la cantidad de ficheros de los que se dispone (relacionado con el balanceo de los datos).
- **Implementación de la estructura TDNN:** Alicia Lozano también ha aportado la implementación de una capa TDNN, ya que ésta no está disponible en Keras. Esta funcionalidad luego ha sido integrada en algunas de las redes desarrolladas en este trabajo de fin de grado.
- **LDA y GPLDA:** También ha aportado el análisis discriminante lineal y el análisis gaussiano discriminante probabilístico lineal, además de funciones auxiliares para la creación de gráficas comparativas de resultados con estas técnicas, como histogramas o funciones de densidad de probabilidad.
- **Conjunto de i-vectors:** Como los modelos de variabilidad total y la consiguiente extracción de i-vectors quedan fuera del ámbito de este trabajo, Alicia Lozano también ha proporcionado el conjunto de i-vectors correspondientes con el dataset empleado para poder así realizar comparativas de resultados.

6.4. Contribuciones

En esta sección se detallan las contribuciones que este trabajo de fin de grado ha realizado al punto de partida aportado por la tutora.

- **Optimización del código:** Los cálculos relacionados con el entrenamiento de una red neuronal están más optimizados en la tarjeta gráfica que en el procesador, por lo que una de las primeras modificaciones del modelo base fue prepararlo para poder ser ejecutado en entornos gráficos. Además, al tener los datos distribuidos en múltiples ficheros las operaciones de carga y preparación de los datos eran costosas, por lo que se modificó el modelo de forma que los cálculos relacionados con el entrenamiento se realizaran en la tarjeta gráfica y de manera paralela la preparación de los datos se realizara en el procesador.
- **Nuevas técnicas y pruebas:** Aunque en trabajos anteriores ya se presentaban las topologías BLSTM y TDNN, así como la división de la red en dos partes, en este trabajo se introduce la aplicación de técnicas que buscan mejorar el rendimiento del sistema y que no habían sido aplicadas anteriormente en modelos similares (como el dropout, la inclusión de ruido gaussiano o el desbalanceado de los datos, entre otros).

- **Adaptación a sistemas distribuidos:** El entorno de pruebas en el que se ha realizado este experimento estaba formado por varios equipos en los que no siempre todos estaban disponibles. Por ello se ha adaptado el código para permitir su ejecución por partes en diferentes equipos de manera paralela y su integración posterior.
- **Mecanismos de recuperación ante fallos:** El entrenamiento de los modelos empleados y la posterior extracción de embeddings son procesos lentos, que pueden durar hasta varios días. Era, por tanto, necesario implementar un mecanismo de copia de seguridad y de recuperación ante posibles fallos en el sistema. El mecanismo desarrollado guarda en disco resultados parciales de ambos procesos, de forma que en caso de caída del sistema sólo es necesario recuperar el proceso desde el último punto guardado.

Glosario de acrónimos

- **DNN**: *Deep Neural Network*
- **TDNN**: *Time Delay Neural Network*
- **RNN**: *Recurrent Neural Network*
- **EER**: *Equal Error Rate*
- **CNN**: *Convolutional Neural Networks*
- **DCT**: *Discret Cosine Transform*
- **BLSTM**: *Bidirectional Long Short-Term Memory*
- **LDA**: *Linear Discriminant Analysis*
- **GPLDA**: *Gaussian Probabilistic Linear Discriminant Analysis*
- **GMM**: *Gaussian Mixture Modelling*
- **UBM**: *Universal Background Model*
- **BLSTM**: *Bidirectional Long Short-Term Memory*
- **MFCC**: *Mel Frequency Cepstral Coefficient*
- **FA**: *Factor Analysis*
- **JFA**: *Joint Factor Analysis*
- **ReLU**: *Rectified Linear Unit*
- **ML**: *Maximum Likelihood*
- **MAP**: *Maximum A Posteriori*
- **PLP**: *Perceptual linear predictive*

Bibliografía

[1] John H.L Hansen and T.Hasan. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal Processing Magazine*, 32:74–99, 2015.

[2] Lianzhang Zhu, Leiming Chen, Dehai Zhao, Jiehan Zhou, and Weishan Zhang. Emotion recognition from chinese speech for smart affective services using a combination of svm and dbn. *Sensors*, 17:1694, 2017.

[3] D. Snyder, D. Garcia-Romero, D. Povey, and D. Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *INTERSPEECH*, pages 999–1003, 2017.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[5] D. Reynols. An overview of automatic speaker recognition technology. *ICASSP*, 4:4072–4075, 2002.

[6] T.Kinnunen and H.Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 115:47–54, 2009.

[7] A.Higgins, L. Bahler, and J. Porter. Speaker verification using randomized phrase prompting. *Digital Signal Processing*, 1:89–106, 1991.

[8] Robert J. Vogt, Brendan J. Baker, and Sridha Sridharan. Modelling session variability in text independent speaker verification. In *Eurospeech/Interspeech: Proceedings of the 9th European Conference on Speech Communication and Technology 2005*, pages 3117–3120. International Speech Communication Association (ISCA), 2005.

[9] Douglas A. Reynols, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1):19–41, 2000.

[10] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEE transactions on speech and audio processing*, 3:72–83, 1995.

[11] D. Reynolds. Comparison of background normalization methods for text-independent speaker verification. *Eurospeech*, pages 963–966, 1997.

[12] J. L. Gauvain and C. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994.

[13] N. Dehak, P. Kenny, R. Dehak, and P.Dumouchel. Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19:788 – 798, 2011.

[14] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artifical neural networks: A tutorial. *Computer*, 29, 1996.

[15] Laurene Fausett, editor. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., 1994.

[16] Geoffrey E. Hinton and Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:2006, 2006.

[17] Geoffrey Hinton, Li Deng, Dong Yu, and George Dahl. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97, 2012.

[18] O. Abdel-Hamid, A. Mohamed, and H. Jiang. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22:1533–1545, 2014.

[19] Awni Y. Hannun, Carl Case, and Jared Casper. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[20] F. Richardson, D. Reynolds, and N. Dehak. A unified deep neural network for speaker and language recognition. In *INTERSPEECH*, 2015.

[21] Alicia Lozano-Diez, Ruben Zazo, Doroteo T. Toledano, and Joaquin Gonzalez-Rodriguez. An analysis of the influence of deep neural network (dnn) topology in bottleneck feature based language recognition. *PLOS ONE*, 12:1–22, 2017.

[22] A. Lozano-Diez, O. Plchot, P. Matejka, and J. González-Rodríguez. Dnn based embeddings for language recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5184–5188, 2018.

[23] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, and Lukas Burget. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

[24] Google Brain Team. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[25] François Chollet et al. Keras api. `https://keras.io`, 2015.

[26] Md. Khademul Islam Molla and Keikichi Hirose. On the effectiveness of mfccs and their statistical distribution properties in speaker identification. *2004 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VCIMS).*, pages 136–141, 2004.

[27] Y. Murphey and H. Guo. Neural learning from unbalanced data. *Applied Intelligence*, 21:117–128, 2004.

[28] Alicia Lozano. *Bottleneck and Embedding representation of speech for Dnn-based language and speaker recognition*. PhD thesis, Universidad Autónoma de Madrid, 2018.

[29] Rubén Zazo. *Exploiting temporal context in speech technologies using LSTM recurrent neural networks*. PhD thesis, Universidad Autónoma de Madrid, 2018.

[30] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, 2015.